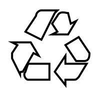


Solaris 公用桌面环境：高级用户和系统管理员指南

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

部件号码: 816–4022–10
2002 年 5 月





020404@3688

内容

序

本手册内容包括定制 Solaris 公共桌面环境 (CDE) 的外观和性能的高级任务。它包括的章节有：

- 定制系统初始化、登录及会话开始
- 添加应用程序，以及为f应用程序及其数据提供接口说明
- 通过网络配置桌面处理、应用程序及数据
- 定制窗口管理、打印、颜色和字体等桌面服务

注意：在此文档中，术语 "IA" 指的是 Intel 32 位处理器体系结构，它包括 Pentium、Pentiun Pro、Pentium II、Pentium II Xeon、Celeron、Pentium III 和 Pentium III Xeon 处理器以及 AMD 和 Cyrix 生产的兼容微处理机芯片。

本书的使用对象

本书的读者包括：

- 系统管理员。本书涉及的许多任务需要根权限。
- 高级用户，他们希望实现用桌面用户接口所无法完成的定制。桌面提供用户自行指定诸多配置文件位置的功能。

阅读本书之前：

用户应该熟悉下面的书籍：

- *Solaris* **公用桌面环境：用户指南**
- *Solaris* **公用桌面环境：用户转换指南**

本书的内容安排

本手册包括下面的章节：

第 1 章 如何配置桌面"登录管理器"的外观和特性。

第 2 章 桌面如何存储和检索会话，以及如何定制会话的启动。

第 3 章 描述 Solaris CDE 启动文件、可能出现的 Solaris CDE 启动故障以及对启动故障的建议解决方案。

第 4 章 "应用程序管理器"如何收集应用程序，并说明如何添加应用程序。

第 5 章 如何为应用程序创建注册软件包。

第 6 章 阐述诸如定制登录配置、设置多重屏幕、网络化桌面和 X 终端、修改用户点文件、 邮件打印定制、桌面环境设置及错误记录类型的高级配置主题。

第 7 章 如何通过网络分发桌面服务、应用程序及数据。

第 8 章 如何添加及删除桌面打印机，以及如何指定缺省打印机。

第 9 章 桌面如何通过网络查找应用程序、帮助文件、图标及其他桌面数据。

第 10 章 介绍操作和数据类型的概念，并说明如何用它们为应用程序提供用户接口。

第 11 章 如何使用"创建操作"应用程序创建操作和数据类型。

第 12 章 如何使用编辑数据库配置文件来创建操作定义。

第 13 章 如何使用编辑数据库配置文件来创建数据类型定义。

第 14 章 如何使用"图标编辑器"，以及桌面图标的命名惯例、 大小和查找路径。

第 15 章 创建新的系统范围的控制和子面板以及定制其他面板。

第 16 章 窗口、鼠标器绑定、键盘绑定和"工作区管理器" 菜单的定制。

第 17 章 如何设置应用程序源，以及桌面如何使用字体和颜色。

第 18 章 运行国际会话过程的系统管理任务。

附录 A 是 dtconfig(1) 手册页的副本。

定购 Sun 文档

Fatbrain.com 是一个 Internet 专业书店，它储有 Sun Microsystems， Inc. 公司精选的产品文档。

有关文档清单以及如何定购，请查阅 Internet 网址 http://www1.fatbrain.com/documentation/sun 的 Fatbrain.com 上 Sun 文档中心。

访问 Sun 联机文档

在 docs.sun.com[SM] Web 站点可以访问 Sun 技术联机文档。您可以浏览 docs.sun.com 归档或查找特定书标题或主题。其 URL 为 http://docs.sun.com。

印刷惯例的含义

下表描述了本书所使用的印刷更改:

表 P–1 印刷惯例

字体或符号	含义	实例
AaBbCc123	命令、文件及目录的名称；计算机屏幕输出	编辑 .login 文件。 使用 ls -a 列出所有文件。 machine_name% 您有邮件。
AaBbCc123	键入的内容，与计算机屏幕输出相对照	machine_name% **su** 口令：

表 P–1 印刷惯例 （续下）

字体或符号	含义	实例
AaBbCc123	命令行占位符：用真实的名称或数值替换	删除一个文件，请键入 **`rm`** *filename*。.
AaBbCc123	书籍的标题、新词或要强调的词。	请参阅 **用户指南** 第六章。 这些称为 *class* 选项。 执行此操作，必须是**超级用户**。

命令实例的 shell 提示

下表显示对 C shell 、Bourne shell 和 Korn shell 的缺省系统提示及超级用户提示。

表 P–2 shell 提示

shell	提示
C shell 提示	`machine_name%`
C shell 超级用户提示	`machine_name#`
Bourne shell 和 Korn shell 提示	`$`
Bourne shell 和 Korn shell 超级用户提示	`#`

第 1 章

配置登录管理器

“登录管理器”是负责显示登录屏幕，验证用户以及启动用户会话的服务器。图形登录因有位图显示而成为传统字符模式登录的有力替代物。 由登录服务器管理的显示器可以直接连接到登录服务器或者连接到网络上的 X 终端或工作站上。

注意：必须以超级用户身份启动、停止或定制登录服务器。

登录服务器：

- 可以无条件地在位图显示器上，或依照请求，在本地和网络位图显示器上显示登录屏幕
- 调节直接连接的字符控制台显示
- 可以显示选择器屏幕，以便用户在网络上其他登录服务器上显示登录屏幕
- 允许对登录服务器的受控访问
- 提供对传统的字符模式登录的访问

由“登录管理器”管理的显示器可以直接连接到“登录管理器”服务器或者连接到网络上的 X 终端或工作站。对于本地显示器，登录服务器会自动启动 X 服务器并显示登录屏幕。对于网络显示器， 例如 X 终端，登录服务器支持 “X 显示器管理器协议” (XDMCP) 1.0， 该协议允许显示器请求登录服务器在显示器上显示登录屏幕。

启动登录服务器

登录服务器通常在系统引导时被启动。它也可以从命令行启动。

- 要将登录服务器设置为在系统引导时启动，请键入 `/usr/dt/bin/dtconfig -e`
  这时登录服务器即可在系统重新引导时自动启动。

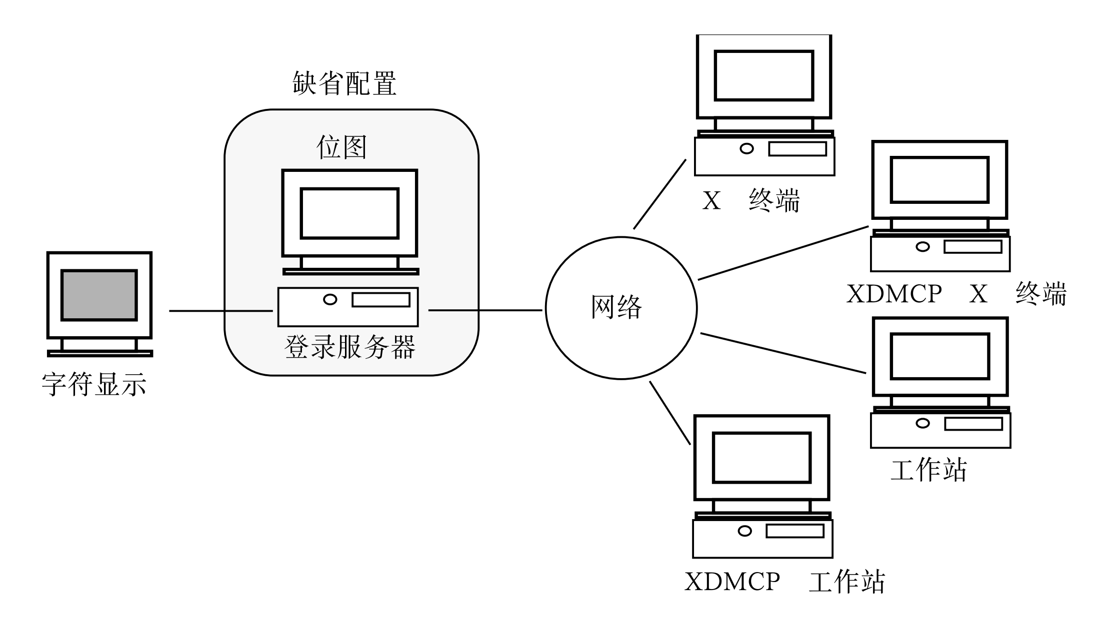
有关桌面配置公用程序 `dtconfig` 的详细信息，请参阅 附录 A. 。它提供了`dtconfig.1`的手册页的副本。

- 要从命令行启动登录服务器，请键入 `/usr/dt/bin/dtlogin -daemon; exit`

注意：虽然从命令行启动登录服务器可用于临时配置测试，但一般应在系统引导时启动登录服务器。

管理本地和网络显示器

图 1-1 显示可供选择的登录服务器配置。



图形 1–1 可供选择的登录服务器配置

查找登录服务器进程 ID

在缺省情况下，登录服务器将其进程 ID 存储于 `/var/dt/Xpid` 中。

要更改它，可以将 `Dtlogin.pidFile` 资源配置在 `Xconfig` 文件中。更改后，登录服务器启动时，指定的目录必须存在。

要修改 `Xconfig`，需将 `Xconfig` 从 `/usr/dt/config` 复制到 `/etc/dt/config` 中。修改 `/etc/dt/config/Xconfig` 之后， 键入下面的内容，以告知登录服务器重新读取 `Xconfig`：

```
/usr/dt/bin/dtconfig -reset
```

这样就发出以下命令 `kill -HUP` *login_server_process_ID*。

例如，要将登录服务器进程 ID 存储于 `/var/myservers/Dtpid`，需将下面的内容放入 `Xconfig`文件：

```
Dtlogin.pidFile: /var/myservers/Dtpid
```

登录服务器重新启动时，它会将其进程 ID 存储于 `/var/myservers/Dtpid`中。登录服务器启动时，`/var/myservers` 目录必须存在。

在本地显示器上显示登录屏幕

启动后，登录服务器检查 Xservers 文件来决定是否需要启动 X 服务器，并决定是否以及如何在本地或网络显示器上显示登录屏幕。

修改 Xservers，需将 Xservers 从 /usr/dt/config 复制到 /etc/dt/config。修改 /etc/dt/config/Xservers 之后，请键入下面的命令行以告知登录服务器重新读取 Xservers：

```
/usr/dt/bin/dtconfig -reset
```

这样就发出以下命令 kill -HUP *login_server_process_ID*

Xservers 行的格式为：

display_name display_class display_type X_server_command

其中

display_name—当连接 X 服务器时，将要使用的连接名告知登录服务器 (下例中的 :0)。* (星号) 的值被扩展为**主机名** :0。指定的编号必须与 *X_server_command* 连接编号中指定的编号相匹配。

display_class—标识该显示器的专用资源 (下例中的本地)。

display_type—告知登录服务器显示器是本地显示器还是网络显示器，以及如何管理登录屏幕上的"命令行登录"选项 (下例中的 local@console)。

X_server_command—标识登录服务器将要用来启动 X 服务器的命令行、连接编号及其他选项 (下例中的 /usr/bin/X11/X:0)。指定的连接编号必须与 *display_name***指定的连接编号一样**。

缺省的Xservers行类似于：

```
:0 Local local@console /usr/bin/X11/X :0
```

在没有本地显示器情况下运行登录服务器

如果登录服务器系统没有位图显示器，则使用 # (磅符号) 为本地显示器注释出 Xservers 行，以便运行不具有本地显示器的登录服务器。例如，

```
# :0 Local local@console /usr/bin/X11/X :0
```

登录服务器启动时，它在后台运行，等待来自于网络显示器的请求。

在本地显示器上访问命令行登录

当用户在登录屏幕上选择"命令行登录"时，登录服务器临时中断 X 服务器，以允许访问在位图显示器终端设备上运行的传统命令行登录。在用户登录及注销完成后，或在指定的超时过后，登录服务器将重新启动 X 服务器。

注意：“命令行登录”选项在网络显示器上不可用。

display_type 控制“命令行登录”的性能。*display_type* 的格式为：

- `local@`*display_ terminal_device*
- `local`
- `foreign`

指定 `local@`*display_terminal_device* 时，登录服务器假定 X 服务器和 `/dev/` *display _terminal_device* 在同一个物理设备上，并且命令行登录 (通常是 `getty`) 正在该设备上运行。当用户选择“命令行登录”时，X 服务器被终止，从而允许访问运行在 `/dev/` *display_terminal_device* 上的命令行登录 (`getty`)。

要禁用显示器上的“命令行登录”选项，需将 `display_terminal_device` 指定为 `none`。缺省 *display_terminal_device* 的值为 `console`。指定 `local` 时，*display _terminal_device* 缺省为 `console`。指定 `foreign` 时，禁用“命令行登录”。

注意：从命令行启动登录服务器时，在本地显示器上禁用“命令行登录”选项。

提供字符显示器控制台

如果登录服务器系统有直接连接的字符显示器作为控制台，可能还要将 *display_terminal _device* 设置为 `none`， 以便禁用位图显示器登录屏幕上的“命令行登录”。

或者，如果命令行登录 (`getty`) 同时运行在字符显示器控制台和位图显示器上，可以将 *display_terminal_device* 更改为位图显示器上的命令行登录 (`getty`) 设备。

例如，如果位图显示器命令行登录 (`getty`) 在设备 `/dev/tty01` 上，则将 *display_type* 更改为 local@tty01。

在网络显示器上显示登录屏幕

登录服务器可以接受来自网络显示器的请求以便在那个特定的显示器上显示登录屏幕。网络显示器通常是 X 终端，也可以是工作站。

为了管理来自网络显示器的请求，登录服务器支持“X 显示器管理器协议” (XDMCP) 1.0 。该协议可以使登录服务器协商并接受或者拒绝来自网络显示器的请求。大多数 X 终端内置 XDMCP。

来自网络显示器的 XDMCP 直接请求

配置 X 终端以使用 XDMCP 直接模式 (查询模式) 时， 将登录服务器主机的主机名告知 X 终端。X 终端引导时，它自动联络登录服务器，登录服务器便在 X 终端上显示登录屏幕。有关讲述如何配置 X 终端以使用 XDMCP 直接模式的信息，请参阅 X 终端文档。

大多数 X 服务器还支持 -query 选项。在此模式中，X 服务器就像 X 终端一样工作，直接联络登录服务器主机并请求它在 X 服务器上显示登录屏幕。例如，在工作站 bridget 上的位图显示器上启动 X 服务器会使登录服务器 anita 在 X 服务器上显示登录屏幕：

```
X -query anita
```

来自网络显示器的 XDMCP 间接请求

配置 X 终端来使用 XDMCP 间接模式时，将登录服务器主机的主机名告知 X 终端。X 终端引导时，它将联络登录服务器，然后登录服务器将在选择器屏幕上给出网络上其他登录服务器主机的列表。从该列表中，用户可以选择主机，该主机将在用户的 X 终端上显示登录屏幕。有关讲述如何配置 X 终端以使用 XDMCP 间接模式的信息，请参阅 X 终端文档。

与直接模式一样，大多数 X 服务器支持 -indirect 选项，该选项使 X 服务器以 XDMCP 间接模式联络登录服务器。

管理非 XDMCP 网络显示器

较早的 X 终端可能不支持 XDMCP 。为了使登录服务器在该类型的 X 终端上显示登录屏幕，要在 Xservers 文件中列出 X 终端的名称。

因为显示器在网络上，所以 *display_name* 将主机名作为名称的一部分。**显示类**可被用来指定特定类的 X 终端的特定资源。(X 终端文档会告诉您 X 终端的显示类。)foreign 这个 *display_type* 告知登录服务器连接现有的 X 服务器，而不是启动它自己的 X 服务器。在这种情况下，有一个 *X_server_command* 没有被指定。

实例

Xservers 文件中的下面的行指示登录服务器在两个非 XDMCP X 终端 (ruby 和 wolfie)上显示登录屏幕：

```
ruby.blackdog.com:0 AcmeXsta foreign wolfie:0 PandaCo foreign
```

控制对登录服务器的访问

在缺省情况下，网络上对登录服务器主机具有访问权的任何主机都可以请求显示一个登录屏幕。可以通过修改 Xaccess 文件来限制对登录服务器的访问。

要修改 Xaccess，需将 Xaccess 从 /usr/dt/config 复制到 /etc/dt/config 中。修改 /etc/dt/config/Xaccess 之后，键入下面的内容以告知登录服务器重新读取 Xaccess：

```
/usr/dt/bin/dtconfig -reset
```

这样就发出以下命令：kill -HUP *login server process ID*。

XDMCP 直接模式

当主机试图通过 XDMCP-direct 连接登录服务器时，该主机名将与 Xaccess 中的各项进行比较，以决定是否允许该主机访问登录服务器。每个 Xaccess 项都是主机名，包括通配符 * (星号) 和 ? (问号)。一个 * (星号) 匹配零个或多个字符，而一个 ? (问号) 与任何单个字符匹配。以 ! (叹号) 开头的项不允许访问；开头没有叹号则允许访问。

例如，Xaccess 包括下面的三项：

```
amazon.waterloo.com
 *.dept5.waterloo.com
 !*
```

第一项允许从主机 amazon.waterloo.com 访问登录服务器，第二项允许从完全域名是以 dept5.waterloo.com 结尾的所有主机访问登录服务器，最后一项不允许任何其他主机访问登录服务器。

XDMCP 间接模式

主机试图通过 XDMCP-indirect 连接登录服务器时，其主机名将与 Xaccess 中的各项进行比较以决定是否允许该主机访问登录服务器。每个 Xaccess 项都类似于 XDMCP-direct 项，包括通配符，只是各项均标以 CHOOSER 字符串。例如：

```
amazon.waterloo.com   CHOOSER BROADCAST
 *.dept5.waterloo.com CHOOSER BROADCAST
 !*  CHOOSER BROADCAST
```

同样，第一项允许从主机 amazon.waterloo.com 访问登录服务器， 第二项允许从完全域名是以 dept5.waterloo.com 结尾的所有主机访问登录服务器，最后一项不允许任何其他主机访问登录服务器。

下列之一可以列在 CHOOSER 之后。

- BROADCAST
- **主机名列表**

BROADCAST 告知登录服务器向登录服务器子网广播，以生成可用登录服务器主机列表。主机名列表告知登录服务器将该表用作可用登录服务器主机列表。例如：

```
amazon.waterloo.com   CHOOSER shoal.waterloo.com alum.waterloo.com
 *.dept5.waterloo.com  CHOOSER BROADCAST
 !*  CHOOSER BROADCAST
```

如果 amazon.waterloo.com 通过 XDMCP-indirect 连接，则提供给它包含 shoal 和 alum 的列表。如果 alice.dept5.waterloo.com 连接，就会给它提供登录服务器子网上的所有可用登录服务器主机的列表。其他 XDMCP-indirect 请求会遭到拒绝。

指定主机名列表的另一方法是定义包括主机名列表的一个或多个宏。例如：

```
%list1     shoal.waterloo.com alum.waterloo.com
 amazon.waterloo.com  CHOOSER%list1
```

检查错误

在缺省情况下，登录服务器将错误记录在 /var/dt/Xerrors 文件中。要改变这种情况，可以将 Dtlogin.errorLogFile 资源放置在 Xconfig 文件中。指定的目录在登录服务器启动时必须存在。

例如，要使登录服务器将错误记录在 /var/mylogs/Dterrors 文件中，需将下面的内容放置在 Xconfig 文件中：

```
 Dtlogin.errorLogFile: /var/mylogs/Dterrors
```

登录服务器重新启动时，登录服务器会把错误记录在 /var/mylogs/Dterrors 文件中。/var/mylogs 目录在登录服务器启动时必须存在。

停止登录服务器

- 要在系统引导时禁用登录服务器启动，需键入：

```
/usr/dt/bin/dtconfig -d
```

这将告知系统在下一次重新引导时不启动登录服务器。

- 要通过中止进程 ID 停止登录服务器，需键入：

```
/usr/dt/bin/dtconfig -kill
```

(这样就发出以下命令：kill *login_server_process_ID*)

注意：中止登录服务器进程将中断由登录服务器管理的所有用户会话。

也可以通过中止进程 ID 以停止登录服务器。登录服务器进程 ID 存储在 /var/dt/Xpid 中或者存储在由 Dtlogin.pidFile 资源在 Xconfig 中指定的文件里。

如果在中止登录服务器的同时登录到了桌面，则桌面会话将立即中断。

登录屏幕

由登录服务器显示的登录屏幕是传统的字符模式登录屏幕的很有吸收力的替代物，而且它所提供的能力超越了字符模式登录提供的能力。



图形 1–2 桌面登录屏幕

与字符模式登录一样，用户输入用户名后再输入口令。它们被验证后，登录服务器为用户启动桌面会话。 用户退出桌面会话时，登录服务器显示新的登录屏幕，进程再次开始。

要定制登录屏幕，可以：

- 更改登录屏幕的外观
- 配置 X 服务器权限
- 更改缺省语言
- 在登录屏幕显示之前发出命令
- 更改登录屏幕“语言”菜单的内容
- 指定命令以启动用户会话
- 在用户的桌面会话启动之前发出命令
- 在用户的会话结束后发出命令

上述的每一项都可以基于所有显示器或基于每个显示器来进行。

更改登录屏幕的外观

要定制登录屏幕的外观，可以更改徽标或图形、欢迎消息和字体。

要修改 Xresources，需将 Xresources 从 /usr/dt/config/*language* 复制到 /etc/dt/config/*language* 中。登录屏幕将在下一次显示时显现出所有更改。要强制登录屏幕重新显示，请在登录屏幕"选项"菜单中选择"重置登录屏幕"。

可由 Xresources 文件的资源规范决定的登录屏幕属性包括：

Dtlogin*logo*bitmapFile—显示为徽标图像的位图或像素图文件

Dtlogin*greeting*persLabelString—个人专用欢迎信息

Dtlogin*greeting*labelString—欢迎消息

Dtlogin*greeting*fontList 欢迎消息的字体

Dtlogin*labelFont 按钮和标号的字体

Dtlogin*textFont 帮助及错误信息的字体

Dtlogin**language* *languageName 语言环境名称的替代文本*language*

▼ 更改徽标

● **设置 Xresources 中的 Dtlogin*logo*bitmapFile 资源。**

徽标可以是彩色像素图或位图文件。

下例使用 Mylogo 位图作为徽标：

```
Dtlogin*logo*bitmapFile: /usr/local/lib/X11/dt/bitmaps/Mylogo.bm
```

▼ 更改欢迎消息

缺省情况下，登录服务器在登录屏幕上显示格式欢迎到**主机名**消息。要更改这个消息：

● **设置 Xresources 中的 Dtlogin*greeting*labelString资源。**

labelString 资源的值可以包含 %*LocalHost*% (将由登录服务器主机名取代)，还包含 %*DisplayName*% (将由 X 服务器显示器名取代)。

下面的实例将欢迎消息更改为"这里是**主机名**!"：

```
Dtlogin*greeting*labelString: 这里是 %LocalHost% ！
```

输入用户名后，登录服务器缺省显示信息欢迎**用户名**的消息。可以通过设置 Xresources 中的 Dtlogin*greeting*persLabelString 资源来更改此消息。persLabelString 的值可以包含 %s (将由**用户名**取代)。

下面的实例将个人专用欢迎消息更改为：您好，**用户名**。

```
Dtlogin*greeting*persLabelString: 您好，%s
```

▼ 更改字体

可以通过设置 Xresources 中的下列的字体资源之一，更改登录屏幕上使用的字体：

Dtlogin*greeting*fontList—欢迎消息的字体

Dtlogin*labelFont—按钮和标号的字体

Dtlogin*textFont—帮助及错误消息的字体

要列出可用的字体，请键入：

xlsfonts [-*options*] [-fn *pattern*]

下面的实例中欢迎消息使用了大字体 (指定的值必须位于同一行中)：

```
Dtlogin*greeting*fontList: -dt-interface \
system-medium-r-normal-xxl*-*-*-*-*-*-*-*-*:
```

▼ 为每种语言提供显示的替代文本

要在登录屏幕“语言”菜单上显示每个语言环境文本而不是缺省显示语言环境名，请在 Xresources 中修改 Dtlogin**language**languageName 资源名的资源：

```
Dtlogin*En_US*languageName: American
```

现在，将显示文本 American，而不显示语言环境名 En_US。

更改登录屏幕性能

要定制登录屏幕性能，可以修改 Xconfig 文件中指定的资源。

要修改 Xconfig，需将 Xconfig 从 /usr/dt/config 复制到 /etc/dt/config 中。修改 /etc/dt/config/Xconfig 之后， 键入下面的内容，以告知登录服务器重新读取 Xconfig：

```
/usr/dt/bin/dtconfig -reset
```

(这样就发出以下命令：kill -HUP *login server process ID*)

在 Xconfig 文件中指定的资源包括：

Dtlogin*authorize—Xaccess 文件说明

Dtlogin*environment—X 服务器环境

Dtlogin*language—缺省语言

Dtlogin*languageList—登录屏幕"语言"菜单的语言列表

Dtlogin*resources—Xresources 规范

Dtlogin*setup—Xsetup 文件规范

Dtlogin*startup—Xstartup 文件规范

Dtlogin*session—Xsession 文件规范

Dtlogin*failsafeClient—Xfailsafe 脚本规范

Dtlogin*reset—Xreset 脚本规范

Dtlogin*userPath—Xsession 和 Xfailsafe 的 PATH

Dtlogin*systemPath—Xsetup、Xstartup 和 Xfailsafe 的 PATH

Dtlogin*systemShell—Xsetup、Xstartup 和 Xfailsafe 的 SHELL

Dtlogin.timeZone—所有脚本的"时区"

更改每个显示器的登录屏幕性能

在下面的例子中，更改 Xconfig 资源就会更改所有显示器登录屏幕的性能。列出的带 * (星号) 的资源可以基于每个显示器来指定。这样就能够为某些显示器指定定制的登录屏幕性能。要为特定显示器指定资源，需将该资源指定为 Dtlogin**displayName* **resource*。例如，如果要为显示器 expo:0 关闭基于用户的访问控制，又要为其他显示器保留访问控制，应指定：

```
Dtlogin*expo_0*authorize: False
```

注意：显示器名中的任何特殊字符，例如 : (冒号) 或 . (句号)，都被替换为 _ (下划线)。

更改 X 服务器访问

在缺省情况下，登录服务器允许基于每个用户的 X 服务器访问控制，并且以在 *HomeDirectory* /.Xauthority 文件中存储并受保护的授权数据为根据。只允许可读取该文件的用户连接 X 服务器。一般来说，这是 X 服务器访问控制的首选方法。

基于用户的访问控制的一种替代方法是基于主机的访问控制。使用这种方法时，如果授权主机对 X 服务器进行访问， 那么该主机上的任何用户均可以连接 X 服务器。使用基于主机的控制的原因如下：

- 较早的 R2 和 R3 X 客户无法使用基于用户的访问控制连接 X 服务器。
- 在无安全保护的网络上，窥探者可能会截取网络上的 X 客户和 X 服务器之间传递的授权数据。

  Xconfig Dtlogin*authorize 资源告知登录服务器使用基于用户的 X 服务器访问控制。要使用基于主机的访问控制，请将授权资源值更改为 False，例如：

  ```
  Dtlogin*authorize: False
  ```

▼ 更改 X 服务器环境

在 X 服务器被登录服务器启动时，如果想为 X 服务器提供一个或多个环境变量和值，可以使用 Xconfig 中的 Dtlogin*environment 资源来指定它们。例如：

```
Dtlogin*environment: VAR1=foo VAR2=bar
```

将使变量 *VAR1* 和 *VAR2* 可用于本地 X 服务器进程。这些变量还将被输出到 Xsession 和 Xfailsafe 脚本中。

▼ 更改缺省语言

当用户从登录屏幕登录到桌面时，用户会话在从“选项”菜单的“语言”子菜单中选择的语言环境下运行。如果用户未选择语言，则使用登录服务器的缺省语言。可以通过设置在 Xconfig 中的 Dtlogin*language 资源来控制缺省语言的值。例如：

```
Dtlogin*language: Ja_JP
```

检查系统文档以决定在系统中安装的语言。

▼ 更改登录服务器语言菜单的内容

在缺省情况下，登录服务器创建包含安装在系统上的所有语言环境的列表的登录屏幕“语言”菜单。用户从登录屏幕语言列表中选择语言环境后，登录服务器将以选定的语言环境重新显示登录屏幕。当用户随后登录时，登录服务器将为该语言环境的用户启动桌面会话。

可以通过修改在 Xconfig 中的 Dtlogin*languageList资源来指定自己的语言列表
：

```
Dtlogin*languageList: En_US De_DE
```

此时，登录服务器在登录屏幕“语言”菜单中只显示 En_US 和 De_DE。

在登录屏幕出现之前发出命令

启动 X 服务器但登录屏幕尚未出现时，登录服务器运行 Xsetup 脚本。Xsetup 以超级用户的权限运行并发出需要在登录屏幕显示之前运行的命令。

修改 Xsetup，需将 Xsetup 从 /usr/dt/config 复制到 /etc/dt/config 中。下一次显示登录屏幕时，将运行修改过的 Xsetup。

启动用户会话之前发出命令

从用户输入了用户名和口令并且它们已通过认证之后到启动用户会话之前，登录服务器运行 Xstartup 脚本。Xstartup 以超级用户的权限运行并发出需要在用户会话启动之前作为超级用户来运行的命令。

修改 Xstartup，需将 Xstartup 从 /usr/dt/config 复制到 /etc/dt/config 中。用户下一次登录时，将运行修改过的 Xstartup。

启动桌面会话

在缺省情况下，登录服务器通过运行 Xsession 脚本来启动用户会话。Xsession 以用户的权限运行并发出启动桌面所需要的命令。

注意：禁止直接更新 Xsession 脚本。

有关如何定制用户桌面会话的启动，请参阅第 2 章。

启动失败安全过程

如果用户从登录屏幕"选项"菜单的"会话"子菜单中选择"失败安全过程"，则登录服务器运行 Xfailsafe脚本。Xfailsafe 以用户的权限运行并发出启动最小的开窗口环境 (通常是"终端"窗口和可选的窗口管理器) 所需要的命令。

修改 Xfailsafe，需将 Xfailsafe 从 /usr/dt/config 复制到 /etc/dt/config 中。用户下一次登录时，将运行修改过的 Xfailsafe。

用户会话结束后

用户退出桌面或失败安全会话之后，登录服务器运行 Xreset 脚本。Xreset 以超级用户权限运行并发出需要在用户的会话结束之后作为超级用户来运行的命令。

要修改 Xreset，需将 Xreset 从 /usr/dt/config 复制到 /etc/dt/config 中。用户下一次登录时，将运行修改过的 Xreset。

登录服务器环境

登录服务器提供一个输出到 Xsetup、Xstartup、Xsession、Xfailsafe 和 Xreset 脚本的环境。此环境在表 1–1中讲述。其他它变量也可由登录服务器输出。

表 1–1 登录服务器环境

环境 变量	Xsetup	Xstartup	Xsession	Xreset	说明
LANG	X	X	X	X	缺省或选定的语言
XAUTHORITY	X	X	X	X	供替换的 X 权限文件 (可选)
PATH	X	X	X	X	Dtlogin*userPath资源 (Xsession、Xfailsafe) 或Dtlogin*systemPath资源 (Xsetup、Xstartup、Xreset) 的值
DISPLAY	X	X	X	X	X 服务器连接编号
SHELL	X	X	X	X	在 /etc/passwd (Xsession、Xfailsafe) 或 Dtlogin*systemShell 资源 (Xsetup、Xstartup、Xreset) 中指定的 shell
TZ	X	X	X	X	Dtlogin.timeZone 资源的或从系统中决定的时区值
USER		X	X	X	用户名
HOME		X	X	X	在 /etc/passwd 中指定的起始目录
LOGNAME		X	X	X	用户名

更改用户或系统路径

登录服务器在运行 Xsession 和 Xfailsafe 脚本时设置 PATH 环境变量。可以为这些脚本提供替代路径。

▼ 更改用户路径

- **设置 Xconfig 中的 Dtlogin*userPath 资源。例如：**

```
Dtlogin*userPath:/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11
```

▼ 更改系统路径

● **设置 `Xconfig` 中的 `Dtlogin*systemPath` 资源。例如：**

```
Dtlogin*systemPath: /usr/bin/X11:/etc:/bin:/usr/bin:/usr/ucb
```

▼ 更改系统 shell

登录服务器在运行 `Xsetup`、`Xstartup` 和 `Xfailsafe` 脚本时设置 SHELL 环境变量。缺省值为 `/bin/sh`。如果要为这些脚本提供替代 shell，可以设置 `Xconfig` 中的 `Dtlogin*systemShell` 资源。例如：

```
Dtlogin*systemShell: /bin/ksh
```

▼ 更改时区

登录服务器在运行 `Xsetup`、`Xstartup`、`Xsession`、`Xfailsafe` 和 `Xreset` 时设置 TZ (时区) 环境变量。缺省值来自系统，所以通常不需要更改此性能。要为这些脚本提供替代时区，需设置 `Xconfig` 中的 `Dtlogin.timeZone` 资源。例如：

```
Dtlogin.timeZone: CST6CDT
```

管理登录管理器

登录服务器启动时，就会启动一个 `dtlogin` 进程。`dtlogin` 进程读取 `Xconfig` 文件以决定初始的登录服务器配置，并定位其他登录服务器配置文件。然后，登录服务器读取 `Xservers` 文件来查看该文件是否有需要显式管理的显示器，同时读取 `Xaccess` 文件以控制对登录服务器的访问。

如果登录服务器从 `Xservers` 文件中查找到该文件需要管理的本地显示器，登录服务器将按照 `Xservers` 文件中的指令来启动 X 服务器，然后在该显示器上显示登录屏幕。

如果登录服务器从 `Xservers` 文件中查找到该文件需要管理一个网络显示器，它将假定 X 服务器已经以指定的显示器名在运行着，并将在该显示器上显示登录屏幕。

然后，登录服务器将等待来自网络的 XDMCP 请求。

对于每个被管理的显示器，登录服务器首先为显示器创建新的 `dtlogin` 进程。这意味着如果登录服务器正在管理 *n* 个显示器，就会有 *n+1* 个 `dtlogin` 进程。登录服务器将运行 `Xsetup` 脚本，装入 `Xresources` 文件，然后运行 `dtgreet` 来显示登录屏幕。一旦用户输入用户名和口令并获得认证后，登录服务器将运行 `Xstartup` 脚本，然后运行 `Xsession` 或 `Xfailsafe` 脚本。用户退出会话时，登录服务器将运行 `Xreset` 脚本。

如果登录服务器接到 XDMCP-indirect 请求，它将运行 dtchooser 以便在显示器上给出登录服务器主机的列表。当用户从该列表中选择主机时，该主机上的登录服务器将管理显示器。

对于 Xaccess、Xconfig、Xfailsafe、Xreset、*language* /Xresources、Xservers、Xsetup 和 Xstartup 配置文件，登录服务器按缺省设置地首先在 /etc/dt/config 中查找，然后在 /usr/dt/config 中查找，并使用首先查找到的文件。

登录管理器文件

登录管理器文件的缺省位置为：

/usr/dt/bin/dtlogin—登录服务器和显示器管理器

/usr/dt/bin/dtgreet— 显示显示器的登录屏幕

/usr/dt/bin/dtchooser—显示显示器的选择器屏幕

/usr/dt/bin/Xsession—启动桌面会话

/usr/dt/config/Xfailsafe—启动失败安全会话进程

/usr/dt/config/Xconfig—登录服务器配置文件

/usr/dt/config/Xservers—登录服务器显示器说明文件

/usr/dt/config/Xaccess—登录服务器访问说明文件

/usr/dt/config/*language* /Xresources—显示器布局资源

/usr/dt/config/Xsetup—显示器设置文件

/usr/dt/config/Xstartup—会话前启动文件

/usr/dt/config/Xreset—会话后重置文件

/var/dt/Xpid—登录服务器的进程 ID

/var/dt/Xerrors—登录服务器的错误记录文件

第 2 章

配置会话管理器

“会话管理器”负责启动桌面并能够自动保存和恢复运行的应用程序、颜色、字体、鼠标性能、音频音量及键盘击键。

使用“会话管理器”可以：

- 为所有桌面用户定制初始会话
- 为所有桌面用户定制环境和资源
- 更改会话启动消息
- 更改会话启动工具和守护程序的参数
- 为所有用户定制桌面颜色用法

什么是会话?

会话是在用户的桌面上所给出的应用程序、设置及资源的集合。会话管理是一套惯例和协议，它们启用“会话管理器”保存和恢复用户的会话。用户能够登录到系统，并且提供给用户的处于运行状态的应用程序、设置和资源的集合与用户注销时提供给用户的相同。用户第一次登录到桌面时，会装入缺省的初始会话。此后，“会话管理器”便支持当前和起始会话的概念。

初始会话

当用户第一次登录到桌面时，“会话管理器” 将使用系统缺省值生成该用户的初始会话。在缺省情况下，“文件管理器”及“桌面介绍” (帮助卷) 将启动。

当前会话

无论在登录时是从保存的起始会话、保存的当前会话，还是从系统缺省初始会话恢复，用户正在运行的会话总视为当前会话。当用户退出会话时，基于用户的“式样管理器启动”设置，“会话管理器”会自动保存当前会话。用户下次登录到桌面时，“会话管理器”会重新启动先前保存的当前会话，这意味着桌面将恢复到与用户上次注销时相同的状态。

起始会话

还可以让桌面在用户每次登录时都恢复为相同的状态，不管用户注销时桌面是什么状态。用户可以保存当前会话的状态，然后使用“式样管理器启动”设置，使“会话管理器”在用户每次登录时启动该会话。

显示器专用会话

要为特定的显示器运行专用会话，用户可以创建显示器专用会话。若想这样，用户可以将 *HomeDirectory* /.dt/sessions 目录复制到 *HomeDirectory* /.dt/*display* 中，其中*display* 是实际的、无限定的主机名 (例如，pablo:0 是有效的，而 pablo.gato.com:0 和 unix:0 是无效的)。用户在显示器 pablo:0 上登录时，“会话管理器” 将启动该显示器专用会话。

启动会话

“会话管理器”是通过 /usr/dt/bin/Xsession 启动的。当用户使用“登录管理器”登录时，在缺省情况下，启动 Xsession。

或者，用户也可以使用传统的字符模式 (getty) 登录进行登录，并使用启动 X 服务器的工具，例如 xinit，来手动启动“会话管理器”。例如：xinit /usr/dt/bin/Xsession。

当会话启动时

“会话管理器”启动时，它经过下面的步骤启动用户的会话：

1. **查找** *HomeDirectory*/.dtprofile **脚本**
2. **查找** Xsession.d **脚本**

3. **显示欢迎消息**
4. **设置桌面查找路径**
5. **收集可用的应用程序**
6. **随意地查找** *HomeDirectory*/.profile **或** *HomeDirectory* /.login
7. **启动 ToolTalk® 消息传送守护程序**
8. **装入会话资源**
9. **启动颜色服务器**
10. **启动"工作区管理器"**
11. **启动会话应用程序**

    以下各节讲述上述步骤。

查找 .dtprofile 脚本

会话一启动，Xsession 脚本便开始查找用户的 *HomeDirectory*/.dtprofile 脚本。*HomeDirectory* /.dtprofile 脚本是一个 /bin/sh 或 /bin/ksh 脚本，它使用户可以为自己的会话设置环境变量。 有关设置环境变量的详细信息，请参阅第 44 页"附加的会话启动定制" 。

如果 *HomeDirectory*/.dtprofile 脚本不存在，例如当用户第一次登录到桌面时，Xsession 将把桌面缺省文件 sys.dtprofile 复制到 *HomeDirectory* /.dtprofile。

桌面缺省值为 /usr/dt/config/sys.dtprofile。要定制 sys.dtprofile 脚本，需将 sys.dtprofile 从 /usr/dt/config 复制到 /etc/dt/config 中，并编辑新文件。

查找 Xsession.d 脚本

查找 *HomeDirectory*/.dtprofile 脚本之后，Xsession 脚本查找 Xsession.d 脚本。这些脚本用来设置附加的环境变量，并为用户的会话启动可选守护程序。缺省的 Xsession.d 脚本为：

0010.dtpaths—将可定制的桌面查找路径记入文档

0020.dtims—启动可选的输入方法服务器

0030.dttmpdir—创建每个用户、每个会话的临时目录

0040.xmbind—将 $XMBINDDIR 设置为桌面缺省值

在 Xsession.d 中还可能另外附有一些供应商专用脚本。

Xsession 首先查找 /etc/dt/config/Xsession.d 目录中的所有文件，然后是 /usr/dt/config/Xsession.d 目录中的所有文件。

桌面缺省的 Xsession.d 脚本位于 /usr/dt/config/Xsession.d 目录。要定制 Xsession.d 脚本，需将脚本从 /usr/dt/config/Xsession.d 复制到 /etc/dt/config/Xsession.d 中， 并编辑新文件。必须具有执行许可才能执行该任务。

同样，要使 Xsession 自动查找您自己的脚本，需将该脚本复制到 /etc/dt/config/Xsession.d 中。

注意：修改或创建 Xsession.d 脚本时，要确保发出的任何前台命令持续时间不长，因为命令所占用的时间将直接影响会话启动时间。如果前台命令不退出，则会话启动将挂起。如果想要 Xsession.d 脚本中运行的命令在会话期间继续运行，就应该在后台运行它们。

显示欢迎消息

查找 *HomeDirectory*/.dtprofile 和 Xsession.d 脚本之后，Xsession 显示覆盖屏幕的欢迎消息。可以定制显示的欢迎消息，或完全关闭消息。dthello 客户机被用来显示消息。

要更改信息文本，可以通过修改 dtstart_hello[0] 变量来更改 dthello 选项。

要更改 dtstart_hello[0]，需创建设置新值的 /etc/dt/config/Xsession.d 脚本。要为所有用户显示当日的消息，需创建可执行的 sh 或 ksh 脚本 (例如 /etc/dt/config/Xsession.d/myvars)，并将 dtstart_hello[0] 设置如下：

```
dtstart_hello[0]="/usr/dt/bin/dthello -file /etc/motd &"
```

类似地，用户可通过设置 *HomeDirectory*/.dtprofile 中的 dtstart_hello[0] 来更改它们的会话的欢迎消息。

要关闭欢迎消息，需设置 dtstart_hello[0]=" "。

有关 dthello 的详细内容，请参阅 dthello 手册页。

设置桌面查找路径

桌面查找路径由 dtsearchpath 在登录时创建。dtsearchpath 使用两类环境变量：

输入变量—全系统的及个人的环境变量，它们的值由系统管理员或最终用户设置。

输出变量—由 dtsearchpath 创建并赋值的变量。每个变量的值都是桌面会话的查找路径。

要更改 dtsearchpath 的命令行选项，需修改 dtstart_searchpath 变量。要为所有用户更改 dtstart_searchpath 变量，需创建可执行的 sh 或 ksh 脚本 (例如 /etc/dt/config/Xsession.d/myvars)，并将 dtstart_searchpath设置如下：

```
dtstart_searchpath="/usr/dt/bin/dtsearchpath"
```

类似地，用户可以通过设置 *HomeDirectory*/.dtprofile 中的 dtstart _searchpath 来只为自己的会话更改 dtsearchpath 选项。

有关 dtsearchpath 的详细信息，请参阅第 9 章。有关 dtsearchpath 选项的详细信息，请参阅 dtsearchpath 手册页。

收集可用的应用程序

设置桌面查找路径后的下一步是使用 dtappgather 收集可用的应用程序。更改 dtappgather 的命令行选项，需修改 dtstart_appgather 变量。要为所有用户更改 dtstart_appgather 变量，需创建可执行的 sh 或 ksh 脚本 (例如 /etc/dt/config/Xsession.d/myvars)，并将 dtstart_appgather 设置如下：

```
dtstart_appgather="/usr/dt/bin/dtappgather &"
```

类似地，通过设置 *HomeDirectory*/.dtprofile 中的 dtstart_appgather，用户可以仅改变自己会话的 dtappgather 选项。

有关 dtappgather 选项的详细信息，请参阅 dtappgather(4) 手册页。

可选地查找 .profile 或 .login 脚本

Xsession 能够查找用户的传统 *HomeDirectory* /.profile 或 *HomeDirectory* /.login 脚本。在缺省情况下禁用该能力。要告知 Xsession 查找 .profile 或 .login 脚本，需将 DTSOURCEPROFILE 设置为 true。

要为所有用户更改 DTSOURCEPROFILE，需创建设置新值的 /etc/dt/config/Xsession.d脚本。为所有用户将 DTSOURCEPROFILE 设置为 true，需创建可执行的 sh 或 ksh 脚本，例如 /etc/dt/config/Xsession.d/myvars，并将 DTSOURCEPROFILE 设置如下：

```
DTSOURCEPROFILE=true
```

类似地，通过在 *HomeDirectory*/.dtprofile 中将 DTSOURCEPROFILE 设置为 true，用户可以为自己的会话更改 DTSOURCEPROFILE。

启动 ToolTalk 消息守护程序

ToolTalk 消息传送守护程序，ttsession，使独立的应用程序可以在相互不直接了解的情况下通信。应用程序创建并发送 ToolTalk 消息来互相通信。ttsession 在网络上通信以传送消息。

要更改 ttsession 的命令行选项，需更改 dtstart_ttsession 变量。要为所有用户更改 dtstart_ttsession 变量，需创建可执行的 sh 或 ksh 脚本 (例如 /etc/dt/config/Xsession.d/*myvars*)，并将 dtstart_ttsession 设置如下：

```
dtstart_ttsession="/usr/dt/bin/ttsession -s"
```

类似地，通过设置 *HomeDirectory*/.dtprofile 中的 dtstart_appgather，用户可以仅改变自己会话的 dtappgather 选项。

有关 ttsession 选项的详细信息，请参阅 ttsession 手册页。有关 ttsession 的详细信息，请参阅*Common Desktop Environment: ToolTalk Messaging Overview*。

启动会话管理器客户机

此时，Xsession 启动 /usr/dt/bin/dtsession，使会话启动进程继续下去。

装入会话资源

"会话管理器"使用 X服务器的 RESOURCE_MANAGER 属性以使桌面资源可用于所有应用程序。"会话管理器"通过下面的方法装入 RESOURCE_MANAGER ：

- 装入系统缺省资源
- 合并由系统管理员指定的所有全系统的资源
- 合并所有用户指定的资源

桌面缺省资源可以在 /usr/dt/config/*language*/sys.resources找到。通过 RESOURCE_MANAGER 特性，使这些资源可用于每个用户的会话。不应编辑此文件，因为它会被后续的桌面安装覆盖。

可以通过创建 /etc/dt/config/*language*/sys.resources 来增加系统缺省资源。在此文件中，您可以覆盖所有桌面用户的缺省资源或为所有桌面用户指定附加的资源。因为在会话启动过程中此文件被合并到桌面缺省资源中，所以只有新的或更新过的资源规范才能放到此文件中。通过 RESOURCE_MANAGER 特性，该文件中指定的资源可用于每个用户的会话。该文件中指定的资源优先于桌面缺省资源文件中所指定资源。

用户可以使用其 *HomeDirectory*/.Xdefaults 文件增加桌面缺省的和全系统的资源。通过 RESOURCE_MANAGER 特性，将使该文件中指定的资源用于该用户的会话。该文件中指定的资源优先于桌面缺省资源或系统管理员资源文件中所指定资源。

注意：“X 工具包内置工具”公用程序指定将从 RESOURCE_MANAGER 或者从 *HomeDirectory* /.Xdefaults 为应用程序装入资源，但是不可同时从二者装入资源。一般，这意味着会忽略用户的 *HomeDirectory* /.Xdefaults 文件。但是，如上所述，在会话启动时“会话管理器”通过将 *HomeDirectory*/.Xdefaults 合并到 RESOURCE _MANAGER 中来调整前者。如果用户更改 RESOURCE_MANAGER /.Xdefaults，那么在用户调用“重新装入资源”操作之前，所做的更改对新的应用程序是看不到的。“重新装入资源”操作将命令“会话管理器”重新装放带有缺省的、全系统的及用户指定资源的 RESOURCE_MANAGER。这样，对全系统的及个人的资源文件所做的更改就可用于应用程序。

有关详细信息，请参阅：

- 第 257 页“设置应用程序资源”
- dtresourcesfile(4) 手册页

启动颜色服务器

“会话管理器”用作桌面的颜色服务器并提供下面的 dtsession 资源集，这些资源集可用来配置 “会话管理器”。

foregroundColor—控制是否为前景颜色分配像素

dynamicColor—指定是否分配只读颜色

shadowPixmaps—指定是否为顶部阴影或底部阴影分配颜色

colorUse—限制颜色的分配

writeXrdbColors—指定是否在资源数据库中放置 *background 和 *foreground 资源

通过创建 /etc/dt/config/*language* /sys.resources， 并在该文件中指定颜色服务器，为所有用户设置颜色服务器资源。

用户也可以类似地通过在 *HomeDirectory*/.Xdefaults 指定颜色服务器资源来为自己的会话设置颜色服务器资源。

有关设置颜色服务器资源的详细信息，请参阅第 267 页“管理颜色”。

启动工作区管理器

“会话管理器”负责启动“工作区管理器”。在缺省情况下会启动 /usr/dt/bin/dtwm。可以使用 wmStartupCommand 资源指定替代窗口服务器。

可以通过创建 /etc/dt/config/*language* /sys.resources，并用该文件中的 Dtsession*wmStartupCommand 资源为窗口管理器指定全路径名及选项，来为所有用户指定替代窗口管理器。

用户可以类似地通过在 *HomeDirectory*/.Xdefaults 指定 Dtsession *wmStartupCommand 资源只为自己的会话指定替代窗口管理器。

有关“窗口管理器”的详细信息，请参阅第 16 章。

启动会话应用程序

会话一启动，“会话管理器”将重新启动被保存为会话的一部分的任何应用程序。即将被恢复为用户初始会话一部分的系统缺省应用程序集可在 /usr/dt/config/*language* /sys.session 中找到。不应编辑此文件，这是因为在后续的桌面安装过程中，它将被无条件覆盖。

有关详细信息，请参阅 dtsessionfile(4) 手册页。

系统管理员可以将 /usr/dt/config/*language* /sys.session 复制到 /etc/dt/config/*language*/sys.session，并修改后面这个文件，这样即可替换作为用户初始会话的一部分而启动的应用程序集。与资源文件不同，该文件将用作桌面缺省文件的完整替换，所以您可以复制一份系统缺省文件，并做任何必要的修改。

附加的会话启动定制

本节包括：

- 设置环境变量
- 设置资源
- 使用与显示器相关的会话
- 在登录时运行脚本
- 恢复备份会话

▼ 设置环境变量

● **要设置全系统的环境变量，请在** /etc/dt/config/Xsession.d**目录中创建用于设置及导出变量的文件。**

例如，如果创建可执行的 ksh 脚本，/etc/dt/config/Xsession.d/*myvars*, 它包含：

```
export MYVARIABLE="value"
```

那么变量 MYVARIABLE 在下次登录时将被设置到每个用户的环境中。

- **要设置个人环境变量，请设置** *HomeDirectory*/.dtprofile **中的变量。**

例如：

```
export MYVARIABLE="value"
```

在下次登录时会把变量 MYVARIABLE 设置到每个用户的环境中。

注意："会话管理器"不自动读取 .profile 或 .login 文件。然而，可以对它进行配置来使用这些文件，参见 第 41 页"可选地查找 .profile 或 .login 脚本"。

▼ 设置资源

- **要设置全系统的资源，需将资源添加到文件** /etc/dt/config/*language*/sys.resources **中。(可能必须创建该文件。)**

注意： .dtprofile 只支持 /bin/sh 或 /bin/ksh 语法。

例如，如果在 /etc/dt/config/C/sys.resources 中指定：

```
AnApplication*resource: value
```

那么资源 AnApplication*resource在下次登录时，将被设置在每个用户的RESOURCE_MANAGER 属性中。

- **设置个人资源，请将资源添加到文件** *HomeDirectory*/.Xdefaults **中。**

▼ 设置显示器专用资源

可为系统上的所有桌面用户设置显示器专用资源。用户也可设置仅限于自己的会话的显示器专用资源。这样用户就可以根据用户登录到桌面时使用的显示器来指定资源。

- **要为系统上的所有桌面用户设置显示器专用资源，需创建指定显示器专用资源的文件** /etc/dt/config/*language* /sys.resources。
- **要设置个人的显示器专用资源，需在主***HomeDirectory*/.Xdefaults **中指定此资源。**

通过将这些资源放入 cpp 条件语句中来界定它们。DISPLAY_displayname 宏是根据 $DISPLAY 变量的值来定义的。这是通过将所有 . (句点) 和 : (冒号) 字符转换为 _ (下划线)，去掉所有屏幕说明，最后给结果加上 DISPLAY_ 前缀来实现的。

例如 :0 的 $DISPLAY 为 DISPLAY_0；blanco.gato.com:0.0 的 $DISPLAY 则是 DISPLAY_blanco_gato_com_0。结果值可用作会话资源文件中的 cpp 测试的一部分。例如，如果在 /etc/dt/config/C/sys.resources 中指定：

```
Myapp*resource: value

#ifdef DISPLAY_blanco_gato_com_0        Myapp*resource: specialvalue1
       #endif

#ifdef DISPLAY_pablo_gato_com_0        Myapp*resource: specialvalue2
       #endif
```

当用户登录到显示器 blanco.gato.com:0上时，资源 MyApp*resource 将在 RESOURCE_MANAGER 中被设置为 specialvalue1 ；当用户登录到显示器 pablo.gato.com:0 上时设置为 specialvalue2；当用户登录到另一个显示器上时设置为 value。

▼ 更改初始会话的应用程序

可以指定替代应用程序，把它作为用户初始会话的一部分来启动。

1. **将** /usr/dt/config/*language* /sys.session **复制为** /etc/dt/config/*language*/sys.session。

2. **修改新的** sys.session **文件。**

   sys.session 中的每项都显现为：

   ```
   dtsmcmd -cmd command_and_options
   ```

   要把附加应用程序作为用户初始会话的一部分来启动，需以全路径名指定新的 sys.session 项。例如，要把 /usr/bin/X11/xclock 作为用户初始会话的一部分，需将 xclock 项添加到 /etc/dt/config/C/sys.session 中：

   ```
   #
   # Start up xclock...
   #
   dtsmcmd -cmd "/usr/bin/X11/xclock -digital"
   ```

▼ 设置显示器专用会话

用户可设置显示器专用会话，以便将会话调节到特定的显示器。

- **将** *HomeDirectory* /.dt/sessions **目录复制到** *HomeDirectory* /.dt/*display* **，其中** *display* **为实际的、无限定的主机名 (**pablo:0 **是有效的，而** pablo.gato.com:0 **或** unix:0 **无效)。**

  例如，要为显示器 pablo.gato.com:0 创建显示器专用会话：

  ```
  cp -r HomeDirectory/.dt/sessions HomeDirectory/.dt/pablo:0
  ```

  用户下次登录到显示器 pablo.gato.com:0 上时，“会话管理器”将启动该显示器专用会话。

在会话过程启动和注销时执行附加的命令

当用户登录到其桌面会话后，便可指定启动附加的命令。这对于设置未被“会话管理器”保存的 X 设置是有用的。例如，用户可使用 `xsetroot` 来定制根 (工作区) 指针。另一种使用方法 是启动不能被“会话管理器”保存及恢复的应用程序。另一个用途是启动“会话管理器”不能保存和恢复的应用程序。如果应用程序在会话恢复时没有重新启动，用户可以使用这种方法启动客户机。

▼ 会话启动时执行附加的命令

- **创建包含命令** *HomeDirectory* `/.dt/sessions/sessionetc` **的文件。**

  该文件通常是脚本而且必须具有执行许可。应在后台运行在 `sessionetc` 中启动的进程。

注意：不要使用 `sessionetc` 来启动由“会话管理器”自动恢复的客户机。这样的操作可能导致启动应用程序的多个副本。也许不能立即看到副本，因为窗口可能会互相堆叠。

▼ 在注销时执行附加的命令

`sessionetc` 的伴侣文件为 `sessionexit`。在会话退出时使用 `sessionexit` 来执行一些不是由“会话管理器”处理的操作。

- **创建包含命令的文件** *HomeDirectory*`/.dt/sessions/sessionexit`。

  类似于 `sessionetc`，该文件通常是具有执行许可的脚本。

▼ 从备份恢复会话

当“会话管理器”保存会话时，会话信息被存储在 *HomeDirectory*`/.dt/sessions` 目录中；如果使用显示器专用会话，则存储在 *HomeDirectory*`/.dt/`display 目录中。在这些目录中，“会话管理器”创建名为 `current` 或 home 的子目录来分别为相应的当前或起始的会话存储信息。在会话信息被存储之前，“会话管理器”以所创建的名称为前会话做备份，并将它存储在 `current.old` 或 `home.old` 中。

1. **从登录屏幕，通过使用“失败安全会话”或“命令行登录”进行登录。**
2. **将备份会话目录复制到活动名中。例如，要恢复备份起始会话：**

```
    cp -r HomeDirectory/.dt/sessions/home.old
\    HomeDirectory/.dt/sessions/home
```

显示器专用会话可使用相同方式恢复。

▼ 调查会话启动故障

- **检查文件** *HomeDirectory*/`.dt/startlog`。

  “会话管理器”在该文件中记录每个用户的会话启动进度。

会话管理器文件及目录

- `/usr/dt/bin/Xsession`
- `/usr/dt/config/Xsession.d/*`
- `/usr/dt/bin/dtsession`
- `/usr/dt/bin/dtsession_res`
- *HomeDirectory* `/.dt/sessions/current`
- *HomeDirectory* `/.dt/sessions/home`
- *HomeDirectory* `/.dt/`*display* `/current`
- *HomeDirectory* `/.dt/`*display*`/home`

第 3 章

登录和会话启动故障的疑难解答

本章讲述 Solaris CDE 的启动文件和在启动 Solaris CDE 时可能遇到的故障，并且给出启动故障的相应解决方案。

登录启动文件

当 Solaris CDE 登录管理器认证了用户之后，将调用下面的脚本来启动桌面：

```
/usr/dt/bin/Xsession
```

`Xsession` 调用的第一个用户特定的文件是 *HomeDirectory*`/.dtprofile`.

当新用户第一次登录到 Solaris CDE 时，`.dtprofile` 文件将被复制到用户的起始目录中。在缺省情况下，此文件不起任何作用。但此文件中的确包含许多如何可以编辑此文件的说明。用户可以编辑此文件，以便添加指定用户的环境变量。

注意：正如 `Xsession` 脚本调用 `.dtprofile` 一样，此文件使用 `ksh` 语法。

一种有用的编辑是撤销文件 `.dtprofile` 的最后一行的注释：

```
DTSOURCEPROFILE=true
```

此行允许用户的 *HomeDirectory*`/.login` 文件 (用于 `csh` 用户) 或 *HomeDirectory*`/.profile` 文件 (用于其他 shell 用户) 作为启动进程的一部分来查找。

错误记录位置

从 Solaris CDE 登录屏幕的"选项"菜单选择"失败安全过程"，以便绕过正常的 Xsession 启动，来查看错误记录，并且排除可能出现的用户点文件故障。表 3–1 显示错误记录及其位置。

表 3–1 错误记录位置

位置	错误记录
/var/dt/Xerrors	在用户登录之前的 Solaris CDE 登录窗口系统错误
HomeDirectory /.dt/startlog	在调用 Xsession、.dtprofile、.login 或 .profile 期间发生的 Solaris CDE 启动错误
HomeDirectory /.dt/errorlog	在 Xsession 启动之后的 Solaris CDE 错误
HomeDirectory /.dt/sessionlogs	"会话管理器"和"窗口管理器"错误的会话记录所在的目录

有些错误也显示在"系统控制台"窗口中。如果"系统控制台"窗口此时并未运行，则应急控制台记录文件名是 wscon，并与时间／显示码连在一起。例如：

```
/usr/tmp/wsconAAAa004EE:0.0
```

用户启动文件

要寻找 .login (对于 csh 用户) 或 .profile (对于 sh 或 ksh 用户) 文件，需要撤去要读取的文件 .dtprofile 的最后一行的注释为：

```
DTSOURCEPROFILE=true
```

在大多数情况下，这就足够了。但在某些情况下，由于文件 .login 或 .profile 可能含有不适用于 Solaris CDE "登录管理器"的命令，所以您还需要修改该文件。如果其中一条命令出故障，则通常与期待终端键盘输入源 (如 stty、tset 或者与 "wait for input" 命令) 的文件有关。



小心：如果 .dtprofile 被设置为寻找文件 .login 或 .profile，而这些文件中又含有出故障的命令，这些命令将使 shell 崩溃，那么桌面启动将失败。结果是不显现任何桌面。相反，您将看到 Solaris CDE 登录屏幕重新显示。.login 或 .profile 中的启动错误通常将被记录在文件 *HomeDirectory*/.dt/startlog 中。可利用失败安全会话或命令行登录方式来调试在 .login 或 .profile 中出故障的命令。

有关故障及其可能的解决方案的更多信息，请参阅 .dtprofile。出故障的命令一般与终端信息和控制有关。

Solaris CDE 启动实例

本节提供了如何编辑下列的用户启动文件的实例：

- .login (用于 csh 用户)
- .profile (用于 sh 或 ksh 用户)
- .Xdefaults

Solaris CDE 启动进程在 .login 或 .profile 文件的脚本中定义了名为 DT 的 shell 变量，以便在 Solaris CDE 的启动进程中进行检查。这就避免了执行像 tty 和 stty 这样的与终端有关的命令。请参阅下面的脚本实例：

.login (C shell)

```
if ( ! ${?DT} ) then

    stty erase '^h'

endif
```

.profile (sh or ksh)

```
if [ ! "$DT" ]; then

    stty erase '^h'

fi
```

围绕 DT 的外加语法避免了在未定义 DT 时来自 C shell 的警告。当用户利用传统的基于文本的控制台登录提示符来登录时，DT 是未定义的。有关设置点文件的详细信息，请参阅 *HomeDirectory* /.dtprofile 文件。

注意：在 .Xdefaults 中指定 Solaris CDE tty 设置。

在 Solaris CDE 启动时，为用户特定的资源查找用户的 .Xdefaults 文件。例如，ttyModes 确保在像 dtterm 或 xterm 这样的终端仿真窗口中，可以使用用户最喜爱的 tty 设置。下面一行在 .Xdefaults 文件中显示典型的 ttyModes 设置：

```
*ttyModes: erase ^H intr ^C kill ^U start ^Q stop ^S susp ^Z‘
```

注意：最喜爱的 Solaris CDE 资源与缺省的设置不同。

“桌面”的“窗口管理器”的图标放置就是一个缺省设置的实例。在这种情况下，在文件 .Xdefaults 中的下面一行显示了缺省的图标放置设置：

```
Dtwm*iconPlacement: right top
```

第 4 章

添加和管理应用程序

应用程序管理器是用户可用应用程序的桌面容器。

应用程序管理器的结构

顶层的"应用程序管理器"通常包含目录。 每个目录及其内容都被称作**应用程序组**。



图形 4–1 应用程序管理器中的应用程序组

应用程序组及其内容是从本地以及整个网络上的多个位置收集的。

应用程序管理器的目录位置

在文件系统中，"应用程序管理器"是目录 /var/dt/appconfig/appmanager/*login-hostname-display*。用户每次登录时都动态创建此目录。

例如，如果用户 ronv 从显示屏 wxyz:0 登录， 就会创建应用程序管理器目录 /var/dt/appconfig/appmanager/ronv-wxyz-0。

应用程序管理器如何查找和收集应用程序

"应用程序管理器"是通过收集本地和远程的应用程序组而建立的。应用程序组是从位于应用程序查找路径上的目录中收集的。

缺省的应用程序查找路径由表 4–1中所显示的位置组成。

表 4–1 缺省的应用程序查找路径的位置

范围	位置
内置	/usr/dt/appconfig/appmanager/*language*

表 4–1 缺省的应用程序查找路径的位置 （续下）

范围	位置
全系统	/etc/dt/appconfig/appmanager/*language*
个人	*HomeDirectory* /.dt/appmanager

要创建顶层的"应用程序管理器"，登录时会创建从位于应用程序查找路径上的目录中的应用程序组 (目录) 到应用程序管理器目录 /var/dt/appconfig/appmanager/*login-hostname-display* 之间的链接。收集操作是由桌面公用程序 dtappgather 进行的，该公用程序在用户成功登录后由"登录管理器"自动运行。

例如，桌面提供内置应用程序组：

/usr/dt/appconfig/appmanager/*language*/Desktop_Tools

登录时，符号链接创建到：

/var/dt/appconfig/appmanager/*login-hostname-display*/Desktop_Tools

应用程序查找路径可以包括远程目录。这就提供了从位于整个网络上的系统中收集应用程序组的方法。有关详细信息，请参阅第 60 页"向应用程序查找路径中添加应用程序服务器"。

收集应用程序的优先级规则

如果查找路径上存在有复制，则个人应用程序组具有高于全系统组的优先级，全系统组具有高于内置组的优先级。例如，如果 /usr/dt/appconfig/appmanager/C/Desktop_Tools 和 /etc/dt/appconfig/appmanager/C/Desktop_Tools 都存在，要使用的将是 /etc 下的应用程序组。

缺省桌面所提供的应用程序组

非定制的桌面提供四个应用程序组。

- Desktop_Apps
- Desktop_Tools
- Desktop_Controls
- Information
- System_Admin

如何收集应用程序组的实例

图形 4–2显示包含若干不同的应用程序组的应用程序管理器窗口。表 4–2显示这些应用程序组来自的目录。



图形 4–2 典型的“应用程序管理器”窗口

表 4–2 应用程序组来源

名称	所收集的目录
CAD_App	/net/ApServA/etc/dt/appconfig/appmanager/C/CAD_App
DrawingApp	/etc/dt/appconfig/appmanager/C/DrawingApp
Desktop_Apps	/usr/dt/appconfig/appmanager/C/Desktop_Apps
Desktop_Tools	/usr/dt/appconfig/appmanager/C/Desktop_Tools
Information	/usr/dt/appconfig/appmanager/C/Information
OpenWindows	/usr/dt/appconfig/appmanager/C/Information
System_Admin	/etc/dt/appconfig/appmanager/C/System_Admin
MySpreadSheet	/users/anna/.dt/appmanager/MySpreadSheet
Media_Tools	/etc/dt/appconfig/appmanager/C/Media_Tools

如果已经定制了 Information 或 System_Admin 应用程序组，则将另从 /etc/dt/appconfig/appmanager/C收集它们。

CAD_App 组之所以被收集，是因为名为 ApServA 的系统已经添加到应用程序查找路径 (见第 60 页“向应用程序查找路径中添加应用程序服务器”)。 MySpreadSheet 是个人应用程序组，仅供用户 anna 使用。

向应用程序管理器添加应用程序

应用程序添加到应用程序管理器后，应用程序组中会出现一图标，用以启动该应用程序。

许多应用程序提供应用程序组。应用程序组是位于应用程序管理器顶层中的目录，它包含了应用程序的图标以及其他与应用程序有关的文件。

有些应用程序可能没有自己的应用程序组。而是将启动应用程序的图标放置在一个通用应用程序组中。例如， 可创建名为“Games ”的空应用程序组，用作系统安装的所有游戏的容器。

向应用程序管理器添加应用程序的方法

将应用程序添加到应用程序管理器的方法有两种：

- 注册应用程序
- 添加应用程序图标而不注册该应用程序

注册应用程序

应用程序注册提供完整的应用程序集成。

经过注册的应用程序：

- 拥有自己的应用程序组。
- 其桌面配置文件收集在单独一个位置。此组桌面配置文件称作**登记软件包**。
- 可能拥有已注册的帮助卷。

注册应用程序的方法有两种：

- 在安装桌面智能型应用程序时，注册自动进行。请参阅第 58 页“向应用程序管理器中添加桌面智能型应用程序”。
- 现有的应用程序可通过创建登记软件包来注册。请参阅第 58 页“注册现有的或非桌面智能型应用程序”。

登记软件包的使用更加简化了桌面应用程序的管理。登记软件包创建于文件系统中桌面配置文件所在位置之外的其他某个位置。

在不使用登记软件包的情况下添加应用程序

当要使“应用程序管理器”只包含启动应用程序的图标时，这是首选的添加应用程序的方法。

在不使用登记软件包的情况下添加的应用程序：

- 可拥有自己的应用程序组，但通常是将图标放置在现有的应用程序组中
- 将其桌面配置文件直接放置在桌面查找路径下。

请参阅第 58 页"向现有的应用程序组添加应用程序图标"。

▼ 向应用程序管理器中添加桌面智能型应用程序

桌面智能型应用程序是在安装时自动注册到"应用程序管理器"中的应用程序。应用程序的文件集包括了桌面所需要的登记软件包。

1. **使用应用程序所提供的指令来安装应用程序。**
2. **安装完成后，双击 Desktop_Tools 应用程序组中的 "重新装入应用程序"。**
3. **验证安装完成：**
   a. **打开应用程序管理器，检查新的应用程序组是否存在。**
   b. **要打开应用程序，先打开应用程序组，然后双击应用程序的图标。**

▼ 注册现有的或非桌面智能型应用程序

这是将应用程序完全集成到桌面的首选方法。

桌面提供有工具 `dtappintegrate`，它可以在登记软件包文件和桌面搜索路径上的目录之间创建链接。

桌面注册在第 5 章中讲解。

▼ 向现有的应用程序组添加应用程序图标

本步骤讲解如何向现有的应用程序组中添加应用程序图标。

例如，桌面提供名为 System_Admin 的应用程序组，该应用程序组是为与系统管理相关的各种应用程序和脚本而保留的。如果有个用户频繁运行的脚本，您可能想让用户能够通过双击位于 System_Admin 应用程序组中的图标来运行该脚本。

1. **使用"创建操作"为应用程序创建操作定义。**

   有关"创建操作"的详细信息，请参阅第 11 章。

2. **在应用程序组的目录下创建与操作名称相同的可执行文件。 该文件的内容并不相关。**

   例如，如果已创建名称为"Cleanup"的操作，用于运行系统管理工具，则需要创建可执行文件：

   `/etc/dt/appconfig/appmanager/`*language* `/System_Admin/Cleanup`

创建和管理通用应用程序组

通用应用程序是个应用程序组 (目录)， 它不与具体的应用程序产品相关联。例如，内置的应用程序组 Desktop_Tools 是个通用组，它包含许多应用程序的图标，这些应用程序与单独某个产品有关，但并不是单独某个产品的一部分。

可以创建附加的通用应用程序组。例如，您可能想创建称作"游戏"的组来收集系统中可用的各种游戏。

通用应用程序组在范围上可以是全系统，也可以是个人的。

▼ 创建全系统的通用应用程序组

1. **以超级用户登录。**
2. **在** `/etc/dt/appconfig/appmanager/`*language* **下创建目录。**

   目录的名称作为应用程序组的名称。
3. **在 Desktop_Tools 应用程序组中，双击"重新装入应用程序"。**

▼ 创建个人的通用应用程序组

1. **在** *HomeDirectory*`/.dt/appmanager` **下创建目录。**

   目录的名称作为应用程序组的名称。
2. **在 Desktop_Tools 应用程序组中，双击"重新装入应用程序"。**

▼ 定制内置应用程序组

1. **以超级用户登录。**
2. **如果应用程序组位于** `/usr/dt/appconfig/appmanager/`*language*，**则将应用程序组复制到** `/etc/dt/appconfig/appmanager/`*language* **下。**

   例如，下面的命令用于复制 Desktop_Tools 应用程序组：

   ```
   cp -r /usr/dt/appconfig/appmanager/C/Desktop_Tools  \
   /etc/dt/appconfig/appmanager/C
   ```

   应用程序组的新副本将具有高于其内置版本的优先级。
3. **修改应用程序组的副本。例如，可以添加新的操作文件 (和操作具有相同名称的可执行文件)。**

4. **要查看更改，可先注销，然后再重新登录。**

修改用于定位应用程序的查找路径

修改应用程序查找路径的主要原因是为了添加应用程序服务器。把应用程序服务器添加到搜索路径后，“应用程序管理器”收集所有应用程序服务器上的全系统应用程序组。

有关应用程序查找路径的详细信息，请参阅第 137 页“应用程序查找路径”。

缺省的查找路径

缺省的应用程序查找路径所包括的目录列在中。

表 4–3 缺省的应用程序查找路径目录

范围	查找路径目录
个人	*HomeDirectory* /.dt/appmanager
全系统	/etc/dt/appconfig/appmanager/*language*
内置	/usr/dt/appconfig/appmanager/*language*

向应用程序查找路径中添加应用程序服务器

要促成与应用程序服务器的通讯，除了要修改应用程序查找路径， 可能还要执行其他一些配置任务。请参阅 第 122 页“管理应用程序服务”。

▼ 设置全系统的应用程序查找路径

1. **以超级用户登录。**
2. **如果文件** /etc/dt/config/Xsession.d/0010.dtpaths **不存在，则通过复制** /usr/dt/config/Xsession.d/0010.dtpaths **来创建它。**
3. **打开** /etc/dt/Xsession.d/0010.paths **，以便进行编辑。添加或编辑用于设置并输出** DTSPSYSAPPHOSTS **变量行：**

```
export DTSPSYSAPPHOSTS=hostname:[, hostname]
```

例如，下行将系统 ApServA 添加到应用程序查找路径：

```
export DTSPSYSAPPHOSTS=ApServA:
```

4. **通知系统上的所有用户，告诉他们注销后重新登录更改才能生效。**

▼ 设置个人应用程序查找路径

1. **打开** *HomeDirectory* / `.dtprofile`**，以便进行编辑。**

2. **添加或编辑用于设置并输出 `DTSPUSERAPPHOSTS` 变量行：**

   ```
   export DTSPUSERAPPHOSTS=hostname:[，hostname]
   ```

   例如，下行将系统 ApServB 和 ApServC 添加到应用程序查找路径中：

   ```
   export DTSPUSERAPPHOSTS=ApServB:，ApServC:
   ```

3. **注销并重新登录。**

通用应用程序管理器的管理

通用应用程序管理器的管理任务包括：

- 删除应用程序
- 会话期间，重新读取应用程序数据库

▼ 删除应用程序

如果应用程序是使用 dtappintegrate 工具注册的，则还可以使用 dtappintegrate 逆转此过程。如果应用程序被注销，其应用程序组被从应用程序管理器中删除，而其操作、数据类型、图标以及帮助将不再可用。

1. **以超级用户登录。**

2. **运行命令：**

   ```
   dtappintegrate -sapp_root -u
   ```

▼ 在会话期间更新应用程序管理器

如果要添加应用程序并使所作更改立即生效，就必须重建"应用程序管理器"。

● **打开 Desktop_Tools 应用程序组，双击"重新装入应用程序"。**

将应用程序添加到应用程序服务器后，"重新装入应用程序" 对于更新应用程序管理器是有用的。但是，"重新装入应用程序"并不检测已从应用程序服务器删除的应用程序，也不检测已经移动到其他位置的应用程序。用户注销并重新登录后，这些更改才能生效。

更改文本编辑器和终端仿真器

文本编辑器和终端仿真器都可通过在"前面板" 中选择控制，或者通过在"应用程序管理器"中双击图标来启动。

这些应用程序也可以通过其他桌面活动来启动。

- 当用户在"文件管理器"中选择了文本文件，然后在"选中"菜单下选择"打开"时，文本编辑器应用程序就会打开。缺省的文本编辑器是 dtpad。
- 当用户从"文件管理器"的"文件"菜单下选择"打开终端"，或者当操作打开终端仿真器窗口时，终端仿真器就会运行。缺省的终端仿真器是 dtterm。

在上述情况下，可以通过配置桌面来使用不同的文本编辑器或终端仿真器应用程序。

▼ 更改缺省的文本编辑器或终端仿真器

1. **如果更改是全系统的，则以超级用户登录。**

2. **为新的文本编辑器或终端仿真器应用程序创建操作。**

   - 可以使用"创建操作"应用程序。图形 4–3 显示的是一个名为 TextPad 的应用程序填充好的"创建操作"窗口。有关"创建操作"的详细信息，请参阅第 11 章。



图形 4–3 “创建操作”窗口

- **或者**，手动创建操作定义；例如：

```
ACTION TextPad
{
    LABEL                TextPad
    TYPE                 COMMAND
    WINDOW_TYPE          NO_STDIO
    EXEC_STRING          /usr/TP/bin/TextPad % (File)Arg_1%
    DESCRIPTION          Double-click this icon to start the \
                         TextPad application.
}
```

有关手动创建操作定义的信息，请参阅第 12 章。

3. **将包含新操作的配置文件放到适当的目录：**
   - 全系统：/etc/dt/appconfig/types/*language*
   - 个人：*HomeDirectory* /.dt/types

4. **如果它不存在，就创建相应的 user-prefs.dt 文件，其方法是将/usr/dt/appconfig/types/*language*/user-prefs.dt 复制到：**
   - 全系统：/etc/dt/appconfig/types/*language* 目录
   - 个人：*HomeDirectory* /.dt/types 目录

5. **在系统范围或个人的 user-prefs.dt 文件中编辑文本编辑器或者终端的操作。修改 MAP_ACTION 行，使操作映射为新的操作。**

例如，将下行：

```
MAP_ACTION Dtpad
```

更改为：

```
MAP_ACTION TxtPd
```

6. **保存** `user-prefs.dt` **文件。**
7. **在 Desktop_Tools 应用程序组中双击"重新装入应用程序"，以便重新装入操作数据库。**

第 5 章

注册应用程序

本章讲述如何为应用程序创建注册软件包以及如何将应用程序注册到桌面上。

当应用程序完全注册到桌面上后，它具有：

- 自己的应用程序组，位于应用程序管理器的顶层
- 启动该应用程序的操作。此操作由应用程序组中的图标表示
- (可选) 数据文件的数据类型

应用程序的注册是对应用程序的无侵害操作：

- 它不涉及对可执行的应用程序自身的修改。因此，可以在系统中注册现有的应用程序。
- 它无需将应用程序的任何已发送的文件 (如可执行文件和 `app-defaults`) 转移到其他文件位置。
- 可以很容易地逆转。用来注册应用程序的 `dtappintegrate` 工具提供了用于逆转注册过程的命令行选项。

在下列情况下，您会需要创建注册软件包：

- 您是系统管理员，要将现有的应用程序注册到桌面上
- 您是软件程序员，要为桌面智能型应用程序创建安装软件包

应用程序注册概述

本节讲解：

- 应用程序注册的目的
- 应用程序注册为应用程序提供的特性

注意：有关显示如何注册现有的应用程序的详细实例，请参阅第 84 页“创建注册软件包的实例”。

应用程序注册所提供的特性

应用程序注册允许用户采用图形方式来：

- 定位应用程序。

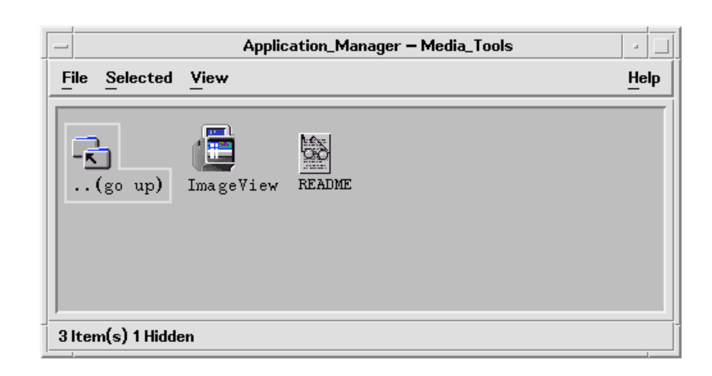
  安装后，应用程序就“注册”到了“应用程序管理器”，并拥有了自己的应用程序组。



图形 5–1 应用程序管理器顶层的应用程序组

- 启动应用程序。

  应用程序的应用程序组含有图标，用户可以双击该图标来启动应用程序。

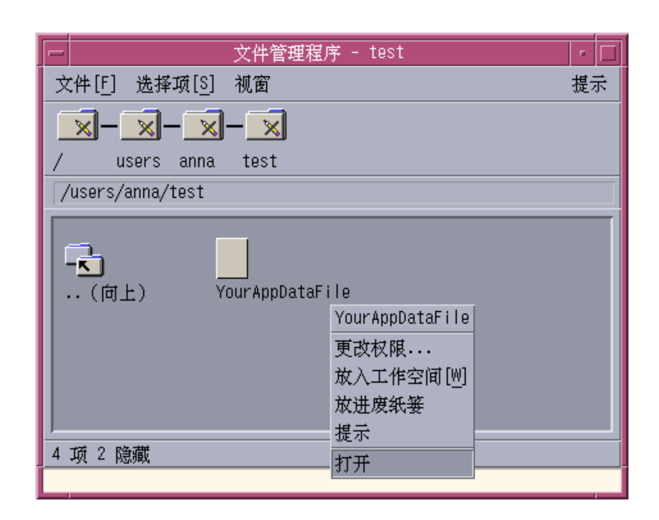


图形 5–2 应用程序组含有用以启动应用程序的图标

- 标识和操作数据文件。应用程序的数据文件会在“文件管理器”中有独特图标。

  用户可以使用数据文件图标来：

  - 启动 (打开) 应用程序
  - 打印数据文件



图形 5–3 数据文件的弹出菜单中包含“打开”和“打印”

- 执行诸如邮寄、压缩、查看或播放 (音频) 数据之类的等其他操作

应用程序注册的目的

注册后的桌面应用程序有某些特定的配置文件，桌面使用这些文件来提供应用程序用户界面：

- 操作和数据类型定义文件
- 图标图像 (像素图或位图) 文件

■ 创建应用程序组的目录和文件
■ (可选) 桌面帮助文件和“前面板”定义文件

为了使桌面识别并使用这些文件，它们必须位于桌面查找路径所指定的某些目录中。

当应用程序的配置文件分散在几个不同的目录中时，对应用程序进行管理会很困难。因此，桌面允许应用程序将所有的桌面配置文件集中到单独一个目录下。这组文件就叫**注册软件包**。

如果应用程序是桌面智能型的，它的安装包本身就带有注册软件包。如果您是自己创建配置文件的系统管理员，则可以亲自创建注册软件包。

由于注册软件包的配置文件不在正确的查找路径目录中，桌面无法使用这些文件。将这些文件放置到适当位置的过程叫**注册**或**集成**应用程序。

桌面提供 `dtappintegrate` 工具，该工具通过在适当的查找路径中创建文件的以符号形式链接的表示来执行注册。

许多桌面智能型应用程序会自动地在安装过程中运行 `dtappintegrate`。如果您是系统管理员，正在集成现有的应用程序，则可以在创建注册软件包后自己运行该工具。

应用程序一旦注册到系统桌面，就可被该系统的所有用户调用。如果该系统被配置为桌面应用程序服务器，此应用程序也可供网络上的其他系统使用。

`dtappintegrate` 工具有一个用于通过中断链接来逆转注册过程的选项。这样，可以很容易地把应用程序从应用程序管理器中删除，以便把应用程序移到另一个应用程序服务器或把它更新。

注册应用程序的一般步骤

注意：有关使用这些步骤来创建应用程序软件包的详细实例，请参阅第 84 页“创建注册软件包的实例”。

---

1. **修改应用程序中任何设置字体和颜色的资源。否则，桌面的动态字体和颜色将不能正常工作。**

   请参阅 第 69 页“第 1 步：修改字体和颜色资源”。

2. **创建应用程序根位置。**

   请参阅 第 70 页“第 2 步：创建桌面应用程序根”。

3. **在应用程序的根下创建目录结构。**

   请参阅第 70 页“第 3 步：创建注册软件包目录”。

4. **创建应用程序的操作和数据类型。**

请参阅第 72 页“第 4 步：创建应用程序的操作和数据类型”。

5. **将帮助文件放置在相应的目录中。**

   请参阅第 75 页“第 5 步：将“帮助文件”放到“注册软件包”中”。

6. **创建应用程序的图标**

   请参阅 第 75 页“第 6 步：创建应用程序的图标”。

7. **创建应用程序的应用程序组。**

   请参阅 第 76 页“第 7 步：创建“应用程序组””。

8. **用`dtappintegrate` 注册应用程序。**

   请参阅 第 81 页“第 8 步：使用 dtappintegrate 注册应用程序”。

第 1 步：修改字体和颜色资源

注意：有关修改应用程序资源的实例，请参阅第 84 页“创建注册软件包的实例”。

桌面提供设置和操作界面字体和窗口颜色的机制。为了使应用程序正确地使用该机制，必须修改应用程序的 `app-defaults` 文件。

修改字体资源

注意：本节适用于使用 OSF/Motif 1.2™ (或更高版本) 创建的应用程序。“式样管理器”无法为用更低版本的 OSF/Motif 编写的应用程序设置界面字体。

如果使用 OSF/Motif 1.2™ (或更高版本) 创建的应用程序没有指定应用程序特定的界面字体，桌面“式样管理器”就会为它设置界面字体。

“式样管理器”提供两种字体：

系统字体—适用于标号、菜单和按钮等系统区

用户字体—适用于文本字段等可编辑区

每种字体的大小都有 7 个级别，在字体对话框中用数字 1 到 7 标记。“式样管理器”的字体通过在 `/usr/dt/app-defaults/`*language* `/Dtstyle` 中的式样管理器资源设置连接到系统中的实际字体。

如果想让应用程序使用“式样管理器”的字体，就应该删除所有指定界面字体的应用程序资源。桌面会自动适当地设置应用程序资源：

`FontList`—设置为系统字体

`XmText*FontList`—设置为用户字体

`XmTextField*FontList`—设置为用户字体

修改颜色资源

"式样管理器"提供动态地更改应用程序颜色能力。应用程序必须是 OSF/Motif 1.1 或 1.2 客户机程序。用其他工具包编写的客户机程序不能动态更改颜色，颜色更改只能在客户机程序重新启动时生效。

使用桌面提供的动态颜色的最简单方法是删除所有用于背景色和前景色的应用程序颜色资源。

第 2 步：创建桌面应用程序根

注意：有关创建应用程序的桌面应用程序根目录实例，请参阅第 84 页"创建注册软件包的实例"的第 2 步。

应用程序的注册软件包文件都被集中到一个被称为应用程序根目录或 *app_root* 的目录下。桌面配置使用的 *app_root* 目录可以和应用程序的安装目录 *app_root* 是同一个目录，也可以是另外某个位置。

例如，假设应用程序安装在 `/usr/BTE` 目录下。此目录可用作桌面配置文件的 *app_root* 目录。但是，如果要集成一个现有的非桌面智能型应用程序，则需要另外创建桌面 *app_root* 目录。这样可以防止在更新应用程序时覆写所创建的配置文件。

例如，系统管理员可以创建 `/etc/desktop_approots/BTE` 作为桌面的 *app_root* 目录。

第 3 步：创建注册软件包目录

注意：有关创建应用程序的注册软件包实例，请参阅第 84 页"创建注册软件包的实例"的第 3 步。

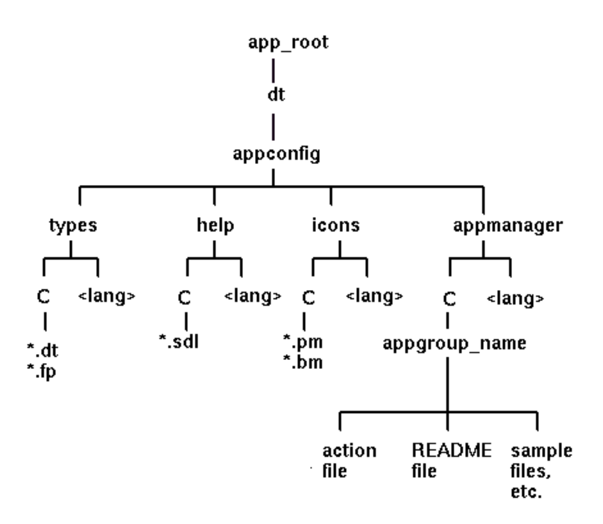
注册软件包是桌面为应用程序提供图形界面所使用的那组桌面配置文件。

注册软件包内容

桌面配置文件包括：

- 操作和数据类型定义文件
- 图标的图像文件
- 应用程序组目录及其内容
- 可选：帮助数据文件和前面板配置文件

  注册软件包聚集在名为应用程序根目录或 *app_root* 的顶层目录下。



图形 5–4 在应用程序根目录下的注册软件包

app_root/dt/appconfig 目录下的配置字段的主要类显示于 表 5–1。

表 5–1 配置区的主要类

子目录	内容
types	操作和数据类型定义文件
help	桌面帮助文件
icons	应用程序的操作和数据类型所使用的位图和像素图图像文件
appmanager	创建应用程序组的目录和内容

每个主要类别都有与语言相关文件的子目录。缺省语言的文件放置在 C 目录中。

▼ 创建注册软件包

● **创建下面的目录。如果要提供与语言相关的配置文件，需要为每种语言创建单独的目录。如果只要提供一种语言，请把文件放置在** `C` **目录中。**

- *app_root* `/dt/appconfig/types/`*language*
- *app_root* `/dt/appconfig/help/`*language*
- *app_root* `/dt/appconfig/icons/`*language*
- *app_root* `/dt/appconfig/appmanager/`*language*`/`*appgroup_name*，其中*appgroup_name* 是应用程序组的名称。

  例如，图形 5–5 显示应用程序管理器，它包含名为"Media_Tools"的 *appgroup_name*。



图形 5–5 在应用程序管理器顶层的应用程序组

`dtappintegrate` 工具只对类型、帮助、图标和应用程序管理器目录下的桌面配置文件进行操作。应用程序的二进制可执行文件、`app-defaults` 和信息种类文件，都单独进行管理。

第 4 步：创建应用程序的操作和数据类型

注意 : 有关创建应用程序操作和数据类型的实例，请参阅第 84 页"创建注册软件包的实例"的第四步。

操作和数据类型为应用程序提供用户界面。

- 操作为启动应用程序的命令提供用户界面
- 数据类型提供用户定制的应用程序数据文件的外观和特性

应用程序所需的操作和数据类型

典型的应用程序需要下面的操作和数据类型定义：

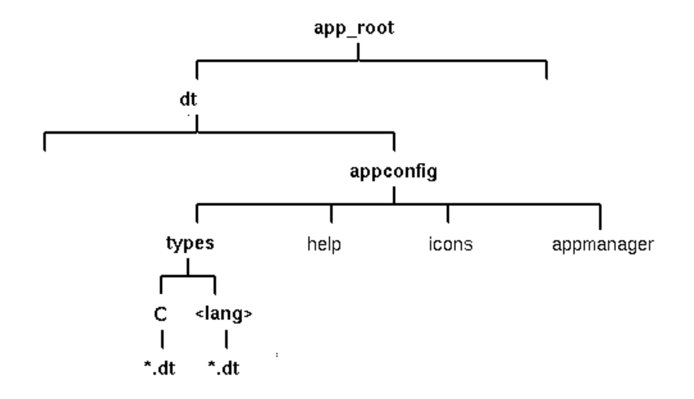
- 打开应用程序的操作。
- 应用程序数据文件的数据类型。如果要创建数据类型，同样也要创建：
  - 应用程序数据文件的“打开”操作
  - 应用程序数据文件的“打印”操作
- 应用程序组的数据类型 (请参阅第 78 页“为应用程序组配置专用图标”)。

  有关桌面中操作和数据类型的使用方法的介绍，请参阅第 10 章。

操作和数据类型定义配置文件的位置

操作和数据类型在配置文件中定义。命名包含操作和数据类型定义的文件的唯一要求是要使其带有 `.dt` 后缀。按照惯例，可以将文件命名为 *action_name* `.dt` 或 *application_name* `.dt`。

将包含操作和数据类型的文件放置在应用程序根目录下的 *app_root* `/dt/appconfig/types/`*language* 目录中。缺省的 *language* 是 `C`。



图形 5–6 操作和数据类型定义文件

创建操作和数据类型的方法

可以采用两种方法为应用程序创建操作和数据类型：

- 使用创建操作工具。

  创建操作工具提供易于使用的界面，它具有供输入用的文本区。然而，该工具有一定的限制。

- 手动创建定义。

  虽然这需要了解创建定义的语法，但它可提供更全面的功能。

▼ 用创建操作工具来创建操作和数据类型

下面的过程采用“创建操作”公用程序为应用程序创建操作和数据类型。

有关“创建操作”的详细信息，请使用它的联机帮助或参阅 第 11 章。

1. **打开 Desktop_Apps 应用程序组，双击“创建操作”。**
2. **使用“创建操作”为应用程序及其数据类型创建操作和数据类型定义。**

   由“创建操作”创建的配置文件会写入 *HomeDirectory* `/.dt/type/` *action_name* `.dt`。其操作文件 (与操作同名的可执行文件) 放置在起始目录中。
3. **用起始目录中创建的操作文件测试操作。**
4. **将操作定义文件** *HomeDirectory* `/.dt/type/` *action_name* `.dt` **复制到** *app_root* `/dt/appconfig/types/`*language* **目录下。**
5. **创建应用程序组目录之后 (请参阅第 76 页“第 7 步：创建“应用程序组””) ，把操作文件** *HomeDirectory*/ *action_name* **复制到** *app_root* `/dt/appconfig/appmanager/`*language*/*appgroup_name* **目录下。**

▼ 手动创建操作和数据类型

● **为应用程序创建包含操作和数据类型定义的配置文件。**

操作和数据类型定义文件的命名必须遵循命名惯例 *name* `.dt`。

可以将操作和数据类型的所有定义放置在一个文件中，也可以分别放置在多个文件内。每个文件使用的文件名都要使系统管理员容易将其与应用程序联系起来。

操作和数据类型名称必须是单词 (没有内嵌的空格)。可以使用下划线字符。 按照惯例，操作或数据类型名的第一个字母要大写。不要使用现有的操作名或文件名。要使用便于高级用户和系统管理员与您的应用程序联系起来的名称。

如果想让应用程序的图标所标记的名称与操作名不同，请让操作定义中包括 `LABEL` 字段。

有关创建操作和数据类型详细信息，请参阅：

- 第 10 章
- 第 11 章
- 第 12 章
- 第 13 章

第 5 步：将“帮助文件”放到“注册软件包”中

注意：有关在注册软件包加入帮助文件的实例，请参阅第 84 页“创建注册软件包的实例”的第 5 步。

如果应用程序包含桌面帮助卷 (用桌面的帮助开发者工具包创建的帮助卷)，帮助卷的主文件 (`*.sdl`) 就应该放在 *app_root* `/appconfig/help/`*language*目录中。

帮助文件使用的图形通常放置在一个图形子目录中。 所有图形必须始终放在创建帮助卷时所建的与主帮助卷 (`*.sdl`) 文件相关的目录下。

在应用程序不提供帮助卷的情况下，如果有帮助开发者工具包，您就可以自行创建一个帮助卷。

帮助卷有两个级别的集成：

- 完全集成。

  当桌面帮助完全集成时，可以从应用程序访问帮助卷—例如，使用项目帮助和“帮助”菜单。完全集成需要修改应用程序的可执行文件。

- 部分集成。

  当桌面帮助部分集成时，可以从帮助管理器的顶层访问帮助卷。但是，无法从应用程序窗口访问帮助卷。另外，您也可以提供从应用程序组访问帮助卷的操作。下面的操作实例显示帮助主文件 `MyApp.sdl` 中的帮助卷：

```
ACTION OpenMyAppHelp
    {
      LABEL           MyAppHelp
      ARG_COUNT       0
      TYPE            COMMAND
      WINDOW_TYPE     NO_STDIO
      EXEC_STRING     /usr/dt/bin/dthelpview -helpVolume MyApp
      DESCRIPTION     Displays help for the MyApp application.
     }
```

第 6 步：创建应用程序的图标

注意：有关创建应用程序图标文件的实例，请参阅第 84 页“创建注册软件包的实例”的第 6 步。

桌面为操作、数据类型和应用程序组提供缺省图标。但是，您可能需要为应用程序创建专用图标。

图标放置在 *app_root* `/dt/appconfig/icons/`*language* 目录中。

桌面所需图标

应用程序在桌面上使用下面的图标图像：

- **操作图标。** 这是让用户双击以启动应用程序 (操作) 的图标。它在启动应用程序操作的 `ICON` 字段引用。

  提供三种尺寸：小图标、中图标和大图标。

- **数据类型图标。** 这是在"文件管理器"中代表应用程序数据文件的图标。它在数据类型定义的 `ICON` 字段引用。

  如果应用程序支持多种数据类型，则应为每种文件类型提供不同的图标。

  提供两种尺寸：小图标和中图标。

- **应用程序组图标。** 这是在应用程序管理器顶层中代表目录的图标。它在应用程序组的数据类型定义中的 `ICON` 字段引用。(请参阅第 76 页"第 7 步：创建"应用程序组""。)

  提供两种尺寸：小图标和中图标。

  为支持彩色 (8 位或 8 位以上) 和单色 (小于 8 位) 显示，每个图标可能需要同时提供像素图和位图两个版本。

表 5–2 图标文件的命名惯例

大小	像素维数	位图名	像素图名
小	16 乘 16	*basename* `.t.bm`	*basename* `.t.pm`
中	32 乘 32	*basename* `.m.bm`	*basename* `.m.pm`
大	48 乘 48	*basename* `.l.bm`	*basename*`.l.pm`

如果未提供位图文件，桌面会将像素图的颜色规范映射为黑白两色。但是这种映射可能不会产生所期望的效果。

有关图标的详细信息，请参阅 第 209 页"图标图像文件"。

第 7 步：创建"应用程序组"

注意：有关创建应用程序组的实例，请参阅第 84 页"创建注册软件包的实例"的第 7 步。

应用程序的操作和数据类型定义一经创建后，还必须创建用于产生用户实际所见对象应用程序组及其内容的配置文件。

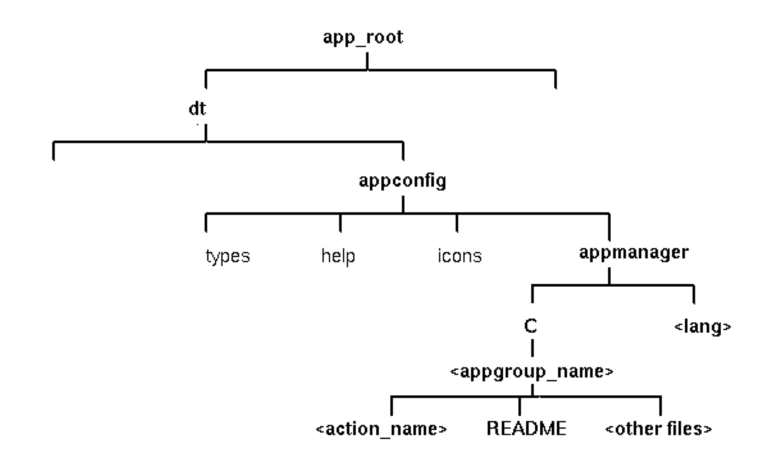
应用程序组是位于应用程序管理器顶层的目录 (请参阅图形 5–1)。

创建应用程序组分为三步：

- 在注册软件包中创建应用程序组目录。
- 可选：为应用程序组配置专用图标。这需要创建应用程序组的数据类型定义。
- 创建应用程序组的内容。

创建应用程序组的目录

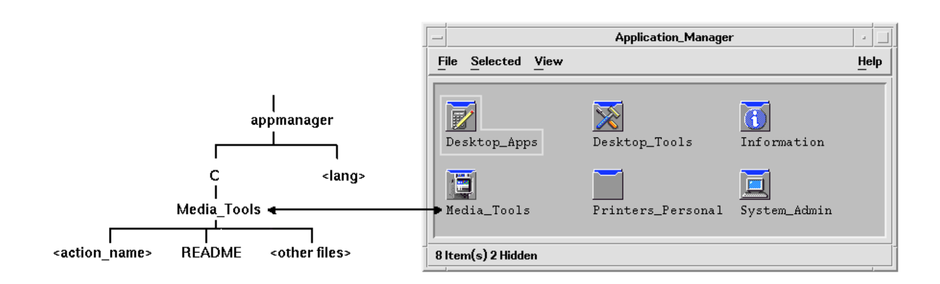
为创建应用程序组，需要在 `appmanager` 下的注册软件包中创建目录，如图形 5–7所示。



图形 5–7 `appmanager` 目录

应用程序组名

图形 5–7 中的 <appgroup_name> 是应用程序组名。



图形 5–8 应用程序组名 (<appgroup_name>)

应用程序组名可以是任何合法的文件 (目录) 名。请使用能描述应用程序功能的名称。

为应用程序组配置专用图标

虽然桌面可以为应用程序组提供缺省图标。 但是，您可能想提供定制图标。

如果想为应用程序组提供专用图标，则必须创建：

- 应用程序管理器顶层目录的数据类型。
- 数据类型的打开和打印操作。

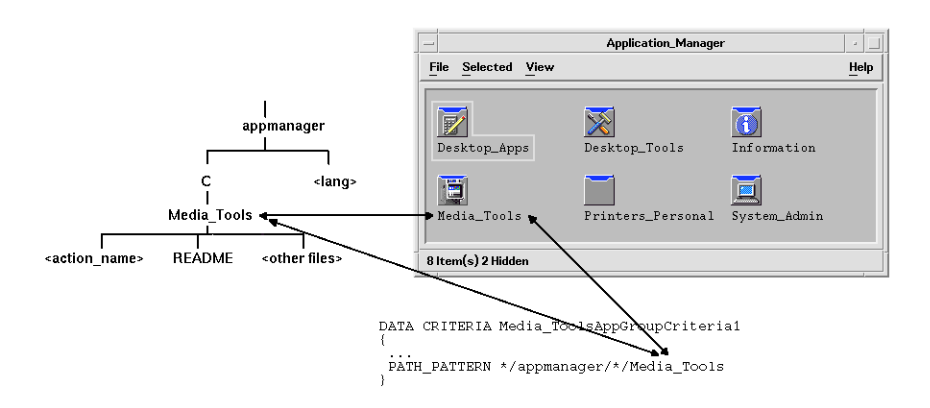
例如，假设想创建名为 Media_Tools 的应用程序组。下面的放置在*app_root* `/dt/appconfig/types/`*language*`/`*name* `.dt` 文件中的数据类型定义指定应用程序组的专用图标。

```
DATA_ATTRIBUTES  Media_ToolsAppgroup
 {
    ACTIONS       OpenInPlace，OpenNewView
    ICON          MediaTools
    DESCRIPTION   Double-click to open the Media_Tools \
                  application group
  }

DATA_CRITERIA  Media_ToolsAppgroupCriteria1
  {
    DATA_ATTRIBUTES_NAME Media_ToolsAppgroup
    MODE                 d
    PATH_PATTERN         */appmanager/*/Media_Tools
  }
```

该定义的属性段指定所用的图标。标准段则指定将数据类型定义到任何名为 `Media_Tools` 并且是 `appmanager` 目录的子目录的目录下。

图形 5–9 显示了应用程序组名和数据类型定义之间的关系。数据类型定义中的 `PATH_PATTERN` 字段将专用图标与应用程序组相连接。



图形 5–9 应用程序组如何获得专用图标

必须为应用程序组数据类型创建“打开”和“打印”操作：

```
ACTION Open
 {
   ARG_TYPE       Media_ToolsAppGroup
   TYPE           MAP
   MAP_ACTION     OpenAppGroup
 }

ACTION Print
 {
   ARG_TYPE       Media_ToolsAppGroup
   TYPE           MAP
   MAP_ACTION     PrintAppGroup
 }
```

OpenAppGroup 和 PrintAppGroup 操作是在 `/usr/dt/appconfig/types/`*language* `/dtappman.dt` 中定义的内置操作。

创建应用程序组的内容

应用程序组中最重要的一项是启动应用程序的图标 (操作图标)。如果应用程序组包括一组应用程序，则通常每个应用程序都有操作图标。

除一个或多个操作图标外，应用程序组还可以包括：

- 一个或多个 README 文件
- 一个或多个样本数据文件
- 模板
- 图标，用户可双击它来查看帮助信息
- 手册页

- 专门的“前面板”控制

  应用程序组可以包含子目录。

创建操作文件 (应用程序图标)

应用程序组应该包含启动应用程序的图标。如果应用程序组提供一组应用程序，则每个程序都应该有图标。这些图标称作**应用程序图标**或**操作图标**，因为它们代表所蕴含的操作。

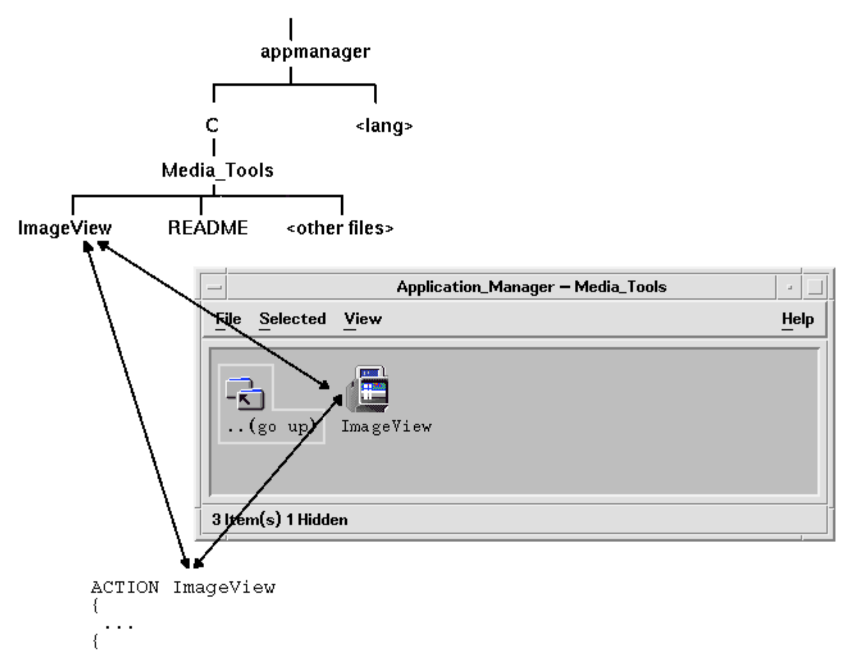
操作图标是通过创建与它将运行的操作同名的可执行文件来创建的：

app_root /dt/appconfig/appmanager/*appgroup_name*/ *action_name*

这个文件被称为**操作文件**，因为它的目的是创建所蕴含的操作的视觉表示形式。

例如，如果已经创建了运行 BestTextEditor 应用程序的，名为 BestTextEditor 的操作，则应该创建名为 BestTextEditor 的可执行文件。在“文件管理器”和应用程序管理器中，操作文件会使用操作定义中指定的图标图像。

图形 5–10 说明操作定义、操作文件和“应用程序管理 器”窗口实际入口之间的关系。



图形 5–10 应用程序图标是应用程序组中的文件

自述文件

桌面提供 README 数据类型供应用程序的 README 文件使用。请使用下列命名惯例之一：

- README
- readme
- README.*
- Read.*.Me
- read.*.me
- READ.*.ME

创建专用前面板

绝大多数情况下，不必提供“前面板”的控制定义；用户可以通过在子面板的“安装图标”控制上放置操作图标，来将应用程序添加到“前面板”中。

如果想使用户可自行安装与操作图标性能不同的控制，例如，该控制监视一个文件并且随被监视文件的改变来改变外观，则可能需要创建包含应用程序控制定义的前面板配置文件。

“前面板”配置文件放置在 *app_root* /dt/appconfig/types/*language* 目录中。命名惯例是*name*.fp。

提供包含控制定义的配置文件后，用户可以通过在子面板的“安装图标”控制上放置 *.fp 文件来将控制添加到子面板。

例如，下面的定义可以放置在应用程序组的“前面板”配置文件中。如果用户将这个文件放置在子面板的“安装图标”控制上，子面板中会创建运行 BestTextEditor 应用程序的单一实例的控制。如果 BestTextEditor 已在运行，该窗口会跳到当前工作区的窗口堆叠的顶部。

```
CONTROL BestTextEditorControl
 {
   TYPE      icon
   ICON      BTEFPanel
   PUSH_RECALL       True
   CLIENT_NAME       BTEd
   PUSH_ACTION       BTEditor
   DROP_ACTION       BTEditor
   HELP_STRING       Starts the BestTextEditor application.
 }
```

有关创建“前面板”配置文件的更多信息，请参阅：

- 第 15 章
- dtfpfile(4) 手册页

第 8 步：使用 dtappintegrate 注册应用程序

注意：有关注册应用程序的实例，请参阅第 84 页“创建注册软件包的实例”的第 8 步。

在应用程序的根目录下一经创建注册软件包后， 即可开始实际的应用程序注册。

应用程序的注册在注册软件包和位于桌面查找路径内的目录之间建立链接 (请参阅第 82 页“dtappintegrate 如何集成应用程序”)。

▼ 用 dtappintegrate 注册应用程序

如果应用程序是桌面智能型的，dtappintegrate 一般作为安装过程的最后一步自动执行。如果它没有自动执行，或者已创建了配置文件以集成非桌面智能型的应用程序，就可以手动运行 dtappintegrate。

1. **以超级用户登录。**
2. **运行下面的命令：**

   ```
   /usr/dt/bin/dtappintegrate -s app_root
   ```

   其中 *app_root* 是桌面应用程序的根目录。有关详细信息，请参阅 dtappintegrate(1) 手册页。

3. **打开 Desktop_Tools 应用程序组，双击“重新装入应用程序”。**
4. **确认应用程序注册正确：**

   a. **显示应用程序管理器的顶层。新的应用程序组应出现在应用程序管理器中。**

   b. **打开应用程序组，双击操作图标。**

dtappintegrate 的语法和选项

```
dtappintegrate -s app_root [-ttarget_path ] [-llanguage ] [-u]
```

-s *app_root*	必要参数，指定已安装应用程序的所在的应用程序根目录。
-t *target_path*	可选参数，缺省为系统位置 /etc/dt/appconfig。指定桌面配置文件链接的位置。指定桌面配置文件链接到哪个位置。必须使用位于应用程序的查找路径中的某个位置。
-l *language*	可选参数，缺省设置为所有语言。指定集成哪一个与语言相关的桌面配置文件。
-u	可选参数，解除应用程序的集成，删除集成期间建立的所有链接。

dtappintegrate 如何集成应用程序

dtappintegrate 的功能是在已安装的文件和桌面查找配置文件的位置之间建立链接。

操作和数据类型

dtappintegrate 创建从注册软件包内的操作和数据类型定义文件到操作数据库帮助查找路径中的全系统目录的符号链接。此操作是通过创建从

app_root/dt/appconfig/types/*language*/*.dt

到

/etc/dt/appconfig/types/*language*/*.dt 的链接来完成的。

帮助信息文件

dtappintegrate 创建从注册软件包内的帮助文件到帮助查找路径中的全系统目录的符号链接。此操作是通过创建从

app_root/dt/appconfig/help/*language*/*help_file*.sdl

到

/etc/dt/appconfig/help/*language*/*help_file*.sdl 的链接来完成的。

图标文件

dtappintegrate创建从注册软件包内的图标文件到图标查找路径中的全系统目录的符号链接。此操作是通过创建从

app_root/dt/appconfig/types/*language*/*.dt

到

/etc/dt/appconfig/types/*language*/*.dt 的链接来完成的。

应用程序组

为把应用程序组放到应用程序管理器的顶层，dtappintegrate 在注册软件包内的应用程序组目录和应用程序查找路径中的全系统目录之间创建符号链接。这是通过创建从目录

app_root /dt/appconfig/appmanager/*language*/*appgroup_name*

到

/etc/dt/appconfig/appmanager/*language*/*appgroup_name* 的链接来完成的。

创建注册软件包的实例

下面的步骤为名为 BestTextEditor 的现有的非桌面智能型应用程序创建注册软件包。

需要了解的有关“BestTextEditor”的信息

本实例假设下列关于 BestTextEditor 应用程序的事实成立：

- 它安装在 /usr/BTE 目录下。
- 用户会话的语言是缺省值 C。
- 启动 BestTextEditor 的命令行是：

  BTEd {*filename*}

  其中 *filename* 是将在新的窗口中打开的数据文件的名称。BestTextEditor 创建自己的窗口—就是说，它不是在终端仿真器窗口中运行。
- BestTextEditor 创建和使用两种类型的数据文件：
  - 文档文件。它们的命名惯例是 *.bte。BestTextEditor 提供了打印它自己的 .bte 数据文件的命令行。该命令的语法是：

    BTEPrint [-d*destination*] [-s]*filename*

    其中：

    -d *destination*—指定目标打印机。

    -s—指定静音打印。应用程序的打印对话框不显示。

    filename—指定要打印的文件。
  - 模板文件。它们的命名惯例是 *.tpl。模板文件不能打印。
- 现有的 BestTextEditor 的非桌面 app-defaults 文件内包含界面字体、前景色和背景色的资源。
- 用帮助开发者工具包创建 BestTextEditorwas 的联机帮助卷。联机帮助卷建立以后，它使用下面的源文件：

  ```
  .../BTEHelp.htg
     .../graphics/BTE1.xwd
     .../graphics/BTE2.xwd
  ```

  同时生成文件 .../BTEHelp.sdl。

注册“BestTextEditor”的步骤

以下是注册 BestTextEditor 的分步过程。

1. **修改字体和颜色资源。**

   在 BestTextEditor 的 app-defaults 文件中，删除设置以下项目的资源：

   - 文本的字体
   - 前景和背景的颜色

2. **创建应用程序的根目录。**

   创建下面的目录：

   ```
   /desktop_approots/BTE
   ```

   如果是集成现有的应用程序，就应该创建与安装地点不同的应用程序根目录；否则，创建的配置文件可能会在更新应用程序时被删除掉。

3. **创建注册软件包目录。**

   创建下面的目录：

   ```
   /desktop_approots/BTE/dt/appconfig/types/C
          /desktop_approots/BTE/dt/appconfig/help/C
          /desktop_approots/BTE/dt/appconfig/icons/C
          /desktop_approots/BTE/dt/appconfig/appmanager/C/ BestTextEditor
   ```

4. **创建应用程序的操作和数据类型。**

   a. **创建定义操作和数据类型的配置文件：**

   ```
   /desktop_approots/BTE/dt/appconfig/types/C/BTE.dt
   ```

   b. **创建运行 BestTextEditor 的操作定义：**

   ```
   ACTION BTEditor
       {
           WINDOW_TYPE      NO_STDIO
           ICON             BTERun
           DESCRIPTION      Double-click this icon or drop\
                            a BTE data file on it to run \
                            BestTextEditor.
            EXEC_STRING      /usr/BTE/BTEd%Arg _1%
        }
   ```

   c. **创建 `*.bte` 文件的数据类型：**

   ```
           DATA_ATTRIBUTES BTEDataFile
       {
          DESCRIPTION       BestTextEditor data file
          ICON              BTEData
          ACTIONS           Open，Print
        }
                   ICON                         BTEData
                   ACTIONS                     Open，Print
            }
   ```

```
DATA_CRITERIA BTEDataFileCriteria1
{
        DATA_ATTRIBUTES_NAME            BTEDataFile
        NAME_PATTERN                        *.bte
        MODE                                    f
}
```

d. 创建 `*.tpl` 文件的数据类型：

```
 DATA_ATTRIBUTES BTETemplateFile
{
   DESCRIPTION     BestTextEditor template file
   ICON            BTETempl
   ACTIONS         Open
 }
          ICON                    BTETempl
          ACTIONS                Open
    }

    DATA_CRITERIAL BTETemplateFileCriteria1
    {
          DATA_ATTRIBUTES_NAME            BTETemplateFile
          NAME_PATTERN                        *.tpl
          MODE                                    f
    }
```

e. 创建 `*.bte` 的打开操作。

```
 ACTION Open
{
   ARG_TYPE      BTEDataFile
   TYPE       MAP
   MAP_ACTION    BTEditor
}
```

f. 创建 `*.bte` 的打印操作。

下面是打印数据文件的简单打印操作。这些操作需要为 LPDEST 环境变量设置一个值，并忽略 `-s` 打印选项。(如果 LPDEST 没有设置，打印操作可能会失败。)

```
 ACTION Print
{
   ARG_TYPE     BTEDataFile
   TYPE      MAP
   MAP_ACTION   BTEPrintData
}

 ACTION BTEPrintData
{
    WINDOW_TYPE        NO_STDIO
   EXEC_STRING        BTEPrint -d $LPDEST%Arg _1%
}
```

下面是 BTEPrintData 操作的另一个版本和伴随脚本。它们一起处理没有设置 LPDEST 或需要静音打印的情况。

```
        ACTION BTEPrintData
    {
       WINDOW_TYPE      NO_STDIO
       EXEC_STRING      /usr/BTE/bin/BTEenvprint \
             %(File)Arg_1%
    }
```

/usr/BTE/bin/BTEenvprint 脚本的内容是：

```
# BTEenvprint
    #!/bin/sh
    DEST=""
    SILENT= ""
    if [ $LPDEST ] ; then
     DEST="-d $LPDEST "
    fi
    BTEPrint $DEST SILENT $1
```

g. 创建 *.tpl 文件的"打开"操作：

```
ACTION Open
    {
        ARG_TYPE         BTETemplateFile
        TYPE          MAP
        MAP_ACTION         BTEditor
    }
```

h. 创建 *.tpl 文件的"打印"操作：

```
        ACTION Print
    {
        ARG_TYPES       BTETemplateFile
        TYPE         MAP
        MAP_ACTION        NoPrint
    }
```

NoPrint 是内置的操作，用于显示对话框，告诉用户文件不能打印。

5. 把帮助文件放入注册软件包内。

a. 把帮助文件放置在下列位置：

```
/desktop_approots/BTE/dt/appconfig/help/C/BTEHelp.sdl
/desktop_approots/BTE/dt/appconfig/help/C/graphics/BTE1.xwd
/desktop_approots/BTE/dt/appconfig/help/C/graphics/BTE2.xwd
```

b. 创建下面的文件：

```
/desktop_approots/BTE/dt/appconfig/types/C/BTEhelp.dt.
```

把下面的操作定义放置在文件内：

```
        ACTION BTEHelp
    {
        WINDOW_TYPE      NO_STDIO
        EXEC_STRING      /usr/dt/bin/dthelpview -helpVolume \
                         BTEHelp.sdl
        DESCRIPTION      Opens the BestTextEditor help volume.
```

```
}
```

6. **创建应用程序图标。**

   用“图标编辑器”创建图标。表 5–3 显示了所要采用的图标大小标准。

   表 5–3 图标大小标准

图标名	大小
basename .t.pm	16 乘 16
basename .m.pm	32 乘 32
basename.l.pm	48 乘 48

   在目录 /desktop_approots/BTE/dt/appconfig/icons/C 下创建下面的图标文件：

   - 代表运行应用程序的操作的图标：BTERun.t.pm，BTERun.m.pm，BTERun.l.pm
   - 代表 *.bte 文件的图标： BTEData.t.pm，BTEData.m.pm，
   - 代表 *.tpl 文件的图标： BTETempl.t.pm，BTETempl.m.pm
   - 代表应用程序组的图标 (用于第 7 步) ：BTEApp.t.pm，BTEApp.m.pm

7. **创建应用程序组。**

   a. **如果尚未创建的话，请创建下面的目录。**

   ```
   /desktop_approots/BTE/dt/appconfig/appmanager/C/BestTextEditor
   ```

   b. **本步骤可选步骤。通过为应用程序组创建一种数据类型和相关联的操作来为程序组提供专用图标。如果忽略这一步，应用程序组将使用缺省的图标。**

   把下面的数据类型和操作定义添加到文件 /desktop _approots/BTE/dt/appconfig/types/C/BTE.dt 中。这些数据类型指定了 BestTextEditor 应用程序组所使用的图标。这些操作提供了与内置应用程序组相同的“打开”和“打印”性能。

   ```
       DATA_ATTRIBUTES BestTextEditorAppGroup
   {
       ACTIONS    OpenInPlace，OpenNewView
       ICON     BTEApp
   }

       DATA_CRITERIA BestTextEditorAppGroupCriterial
   {
       DATA_ATTRIBUTES_NAME BestTextEditorAppGroup
       MODE              d
       PATH_PATTERN         */appmanager/*/BestTextEditor
   }

       ACTION Open
   {
       ARG_TYPE      BestTextEditorAppGroup
   ```

```
    TYPE             MAP
    MAP_ACTION    OpenAppGroup
}

    ACTION Print
{
    ARG_TYPE       BestTextEditorAppGroup
    TYPE           MAP
    MAP_ACTION     PrintAppGroup
}
```

c. 在应用程序组内创建启动应用程序的图标。这需要创建下面的文件：

```
/desktop_approots/BTE/dt/appconfig/appmanager/C \
    /BestTextEditor/BTEditor
```

并使文件可执行。

d. 在应用程序组内创建打开帮助卷的图标。这需要创建下面的文件：

```
/desktop_approots/BTE/dt/appconfig/appmanager/C \
    /BestTextEditor/BTEHelp
```

并使文件可执行。

e. 把其他文件放入应用程序组中；例如，"自述"文件、样本数据和模板文件。

8. 注册应用程序。

在终端仿真器窗口内：

a. 以超级用户登录。

b. 运行下面的命令：

```
/usr/dt/bin/dtappintegrate -s /desktop_approots/BTE
```

c. 打开 Desktop_Tools 应用程序组，双击"重新装入应用程序"。

第 6 章

其他配置

本章讲述设置和系统的管理。

Solaris CDE 的目录结构

本节讲述了桌面环境所包含的主要目录。

/usr/dt

此目录为 Solaris CDE 的安装位置。也可以是一个远程文件服务器的安装点。表 6–1讲述 `/usr/dt` 子目录。

表 6–1 `/usr/dt` 子目录

子目录	说明
`/bin`	SCDE 应用程序和公用程序

表 6–1 /usr/dt 子目录　（续下）

子目录	说明
/lib	SCDE 运行时共享库
/config	缺省系统配置文件
/man	[可选] 手册页
/app-defaults	缺省应用程序资源
/appconfig	缺省应用图标、类型和操作
/examples	[可选] CDE 代码/程序实例
/include	[可选] 开发者包含文件
/palettes	颜色调色板
/share	CDE AnswerBook 文档和缺省背景

/etc/dt

此目录包含定制的工作站专用配置文件。这些文件使您可以用下面的方式来定制环境：

- 设置 X-server 配置选项
- 使用多重屏幕
- 定制工作站操作文件、数据类型、图标和字体

/var/dt

本目录保存了用于诸如"登录管理器"和"应用程序管理器"之类的 Solaris CDE 应用程序的临时文件。

$HomeDirectory

此目录包含与用户的桌面设置有关的用户特定的文件。它们包括对应用程序、颜色搭配方案、工作区菜单和前面板的修改以及一些错误记录。

主要配置文件

对桌面环境的定制大多会涉及许多文件。下面的文件是两个实例：

- /usr/dt/config/Xconfig
- /usr/dt/config/Xservers

Xconfig

Xconfig 是 dtlogin 使用的主配置文件。它设置登录资源并指定 dtlogin 所需的附加文件的位置。以下实例显示的是缺省 Xconfig 文件中的几行：

```
 Dtlogin.errorLogFile:       /var/dt/Xerrors
Dtlogin.servers:            /usr/dt/config/Xservers
Dtlogin*session:            /usr/dt/bin/Xsession
```

有关此文件的详细信息，请参阅 附录 A。您也可以查看文件中的说明。

Xservers

dtlogin，就像它所基于的 XDM (X 显示管理器) 一样，使用 Xservers 文件来指定本地 Xserver 的启动方式。缺省情况下，此文件的最后一行为：

```
:0 Local local@console /usr/openwin/bin/Xsun :0
```

- :0 是指 Xserver 显示为 <localhost:0>。
- local 表示在本地启动新的 X11 服务器。

注意：如果要连接到一个正在运行的 X11 服务器上，将 local 替换为 foreign。

- console 是指"命令行登录"换码为 /dev/console。
- /usr/openwin/bin/Xsun 是到 X11 服务器的路径。

注意：Solaris CDE 和 OpenWindows™ 环境运行同一个 X11 服务器。

有关此文件的详细信息，请参阅 dtlogin(1) 手册页。也可以审阅文件自身的说明。

启动登录服务器

登录服务器通常在系统引导时自动启动。也可从命令行启动登录服务器，但必须首先以超级用户身份登录。

设置登录服务器在系统引导时自动启动：

- **键入下面的命令并按回车键：**

```
# /usr/dt/bin/dtconfig -e
```

这将 `S99dtlogin` 文件添加到 `/etc/rc2.d` 目录中。当重新引导时，登录服务器会自动启动。

禁止登录服务器在系统引导时自动启动：

- **键入下面的命令并按回车键：**

```
# /usr/dt/bin/dtconfig -d
```

从命令行启动登录服务器：

- **键入下面的命令并按回车键：**

```
# /usr/dt/bin/dtlogin -daemon; exit
```

注意：尽管临时配置测试允许从命令行启动登录服务器，但是登录服务器通常还是应该在系统引导时启动。

中止登录服务器、 Xserver 和整个 Solaris CDE 桌面：

- **键入下面的命令并按回车键：**

```
# /usr/dt/bin/dtconfig -kill
```

有关桌面配置公用程序 `dtconfig` 的详细信息，请参阅 附录 A。它提供一份 `dtconfig.1` 手册页的副本。

使用"命令行登录"选项退出 Solaris CDE 登录屏幕：

- **在 Solaris CDE 登录屏幕上"选项"菜单中选择"命令行登录"。等屏幕清除后，按回车键来显示登录提示。**



注意：在完成"命令行登录"后，桌面登录守护程序将自动启动桌面登录屏幕。

启动单个终端仿真窗口：

- **在 Solaris CDE 登录屏幕上从"设置"菜单的"会话"子菜单中选择"失败安全会话"。**



如果想让 Xserver 保持运行状态，可以从"会话"子菜单中选择"失败安全会话"。这将会启动单个 `xterm` 窗口。"失败安全会话"选项总是可用的，即使在"命令行登录"选项不可用时它也可用。

从另一个工作站或网络服务器安装位置来安装已安装的 CDE

如果在工作站上没有足够的硬盘空间，可以从另一工作站或网络服务器安装位置安装已安装的 CDE，因为这种方法不占用该机的硬盘空间。

注意：由于不同的 Solaris 版本和客户工作站之间的 `/usr/dt` 目录结构不同 (SPARC 工作站和 Intel 工作站的目录结构不同)，客户工作站必须安装合适的 NFS 服务器中的 `/usr/dt` 映像。例如一个 Solaris 2.5 的 SPARC 系统，应该从另一个已在 /usr/dt 目录中安装 CDE 的 Solaris 2.5 系统安装它的 /usr/dt 目录。

▼ 安装已安装的 CDE

1. **将以前安装的工作站或网络服务器中的** `/usr/dt` **目录安装到您的工作站的** `/usr/dt` **目录下。**
2. **键入** `/usr/dt/bin/dtconfig-inetd`
3. **要启用 Solaris 桌面登录，请键入：**

```
/usr/dt/bin/dtconfig -e
```

4. **重新引导工作站。**

   有关桌面配置公用程序 `dtconfig` 的详细信息，请参阅 附录 A。它提供一份 `dtconfig.1` 手册页的副本。

▼ 卸载已装配的 CDE 目录

1. **要禁用 Solaris 桌面登录，请键入：**

```
/usr/dt/bin/dtconfig -d
```

2. **键入** `/usr/dt/bin/dtconfig-inetd.ow`
3. **卸载** `/usr/dt`。
4. **重新引导工作站。**

   有关桌面配置公用程序 `dtconfig` 的详细信息，请参阅 附录 A。它提供了一份 `dtconfig.1` 手册页的副本。

配置桌面以使用多个屏幕

标准的登录过程在单个屏幕上启动桌面程序。您也可以编辑 Xconfig 文件在多个屏幕上启动桌面程序。您必须以超级用户身份登录时才可以修改该文件。

注意：若要快速编辑登录配置信息，使用"失败安全会话"来运行单个 xterm 窗口，而不是运行整个桌面程序。

▼ 使桌面在多个屏幕上启动

1. **通过键入以下命令并按回车键，可创建一份 Xserver 的副本：**

```
# cp /usr/dt/config/Xservers /etc/dt/config/Xservers
```

注意： /etc/dt/config/Xservers 覆盖 /usr/dt/config/Xservers。

---

2. **编辑 /etc/dt/config/Xservers 文件来设置两个框架缓冲区 (屏幕)：**

   a. **查找此行：**

```
:0 Local local_uid@console root /usr/openwin/bin/Xsun :0
```

   b. **在这一行末尾添加两个 -dev 选项。本例假设它们被命名为 /dev/fb0 和 /dev/fb1。键入空格后添加以下项：**

```
-dev /dev/fb0 -dev /dev/fb1
```

3. **从"选项"菜单中选择"重置登录屏幕"。**



注意 : 在多个屏幕上运行 Solaris CDE 时，每个屏幕上都显示一个"前面板"。这些"前面板"相互独立，各有独立的配置文件。

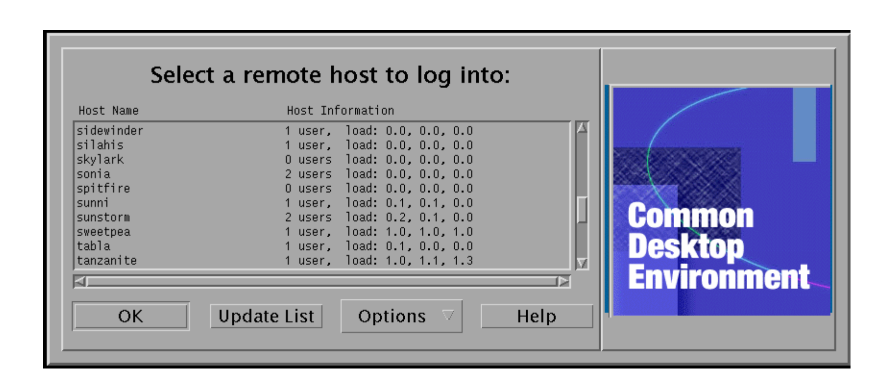
联网的桌面

Solaris CDE 登录管理器对网络敏感。在缺省情况下，登录屏幕会响应 Solaris CDE 选择器的查询。

显示远程主机列表：

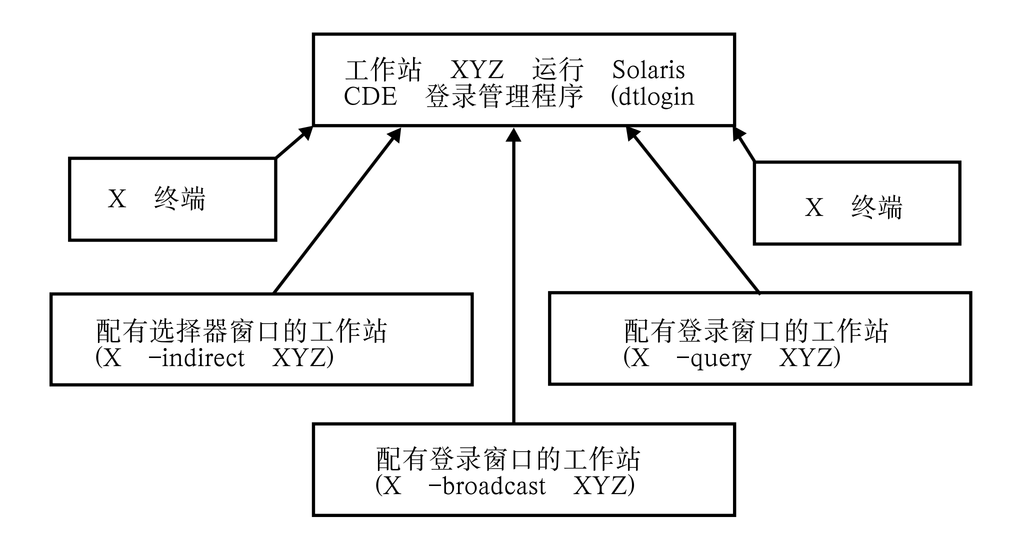
● **在 Solaris CDE 登录屏幕的"远程登录"菜单中选择 "从列表中选择主机"。**

或者不依赖登录屏幕而独立运行选择器，参见第 102 页"将工作站作为 XTerminals 使用"。下面的屏幕显示可用服务器的典型选择器列表。



可以从此列表中选择一个闲置服务器，通过“选择器”来利用它的资源以登录到 Solaris CDE。

联网的任一台运行 Solaris CDE“登录管理器”的工作站都既可以支持一个本地桌面用户，也可以支持多个远程用户，包括把该工作站用作 Solaris CDE 桌面服务器的 Xteminal 用户。(请参见图形 6–1。)

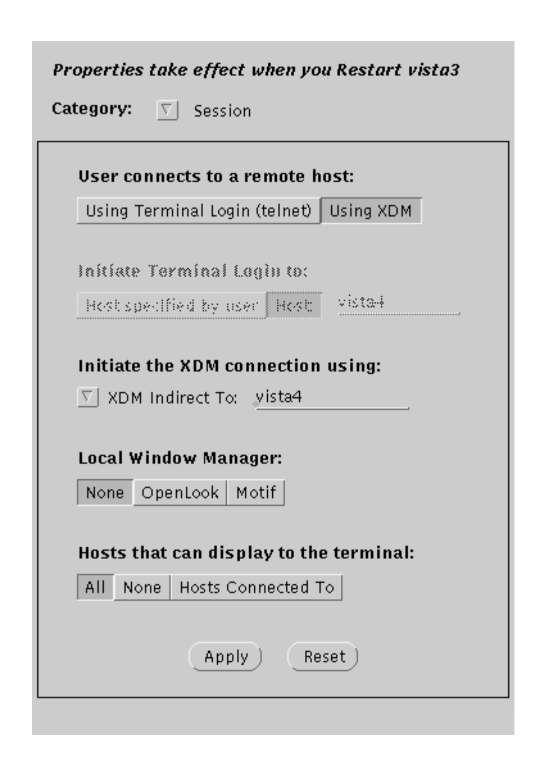


图形 6–1 联网的桌面

注意：图 6-1 可以包括一个混合硬件环境。

使用 XTerminals

任何支持 XDM 协议的 Xterminal 都可以使用"选择器"来登录到 Solaris CDE。要运行"选择器"，请参阅 第 102 页"将工作站作为 XTerminals 使用"。 SPARC Xterminal™ 软件 2.0 版 (或更高版本) 在 Solaris CDE 上运行良好。在 "X 终端控制"窗口上单击"特性"，并选择"会话"作为"类别"来显示如图形 6–2中所示的设置屏幕。

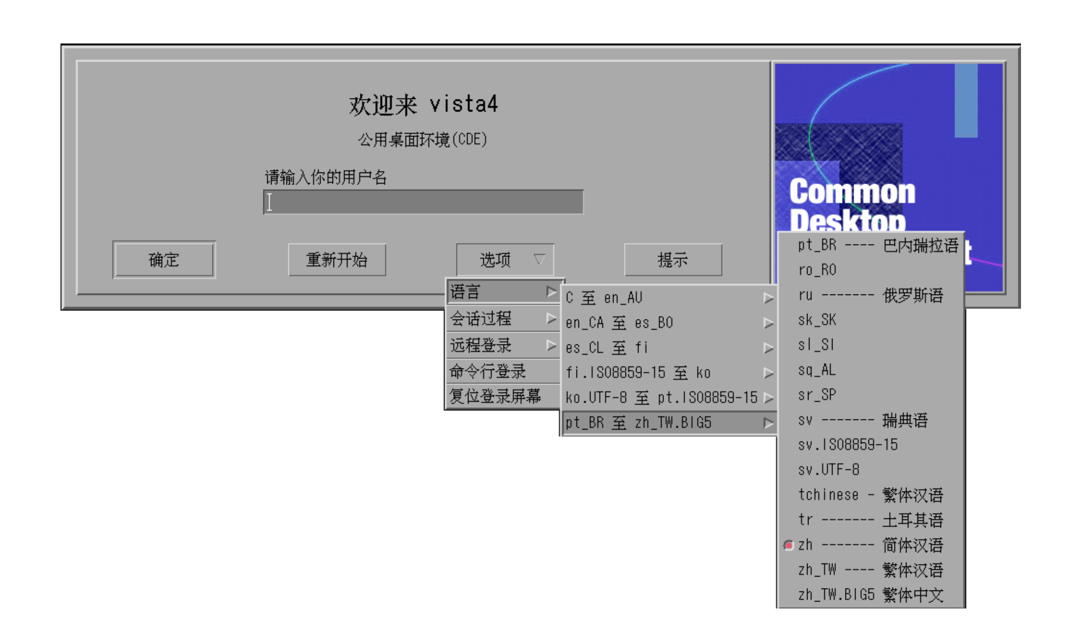


图形 6–2 会话设置屏幕

给"本地窗口管理器"选择"无"，给连接运行 Solaris CDE 的远程系统的 XDM 选择"间接 XDM"。

登录语言环境和字体路径

当登录到 Solaris CDE 上时，选择您的首选语言。下面的 Solaris CDE 登录屏幕显示了"选项"菜单中的"语言"选择。



在工作站上，字体路径被自动设置为包括与首选的语言和系统基本的 “C” 语言环境俩者都相关联的字体 (和假名)。对 Xterminals 来说，这些字体由 X11 字体服务器自动提供。

将工作站作为 XTerminals 使用

在您的工作站比较陈旧，或者工作站的内存等于或少于 16 兆字节的情况下，如果它们安装了 Solaris 2.4 或更高版本的软件 (或是带 Xserver 大型修补程序的 Solaris 2.3)，就可以工作站作为 Xterminal 使用。

注意：如果遵循以下过程而工作站屏幕上的字体显示不正常，则 Solaris CDE 可能没有安装在主工作站上。请使用 `pkgadd` 公用程序从一个 Solaris CDE 1.0.x 的非随附版本中单独添加 `SUNWdtft()` Solaris CDE 字体软件包。

▼ 用选择器来选择主机 CDE 进行登录

1. **退出任何正在运行的窗口系统。**
2. **启动选择器。从控制台的命令行键入下列命令 (使用 `csh`)：**

```
setenv OPENWINHOME /usr/openwin
```

```
/usr/openwin/bin/X -indirectCDE_login_host
```

Xserver 启动，并显示主机的"选择器"窗口。

3. 从"选择器"窗口选择所需的登录主机。

▼ 使用具体的主机 CDE 登录

- **键入下面的命令并按回车键：**

```
/usr/openwin/bin/X -once -queryCDE_login_host
```

Xserver 启动，并显示主机的"登录"屏幕。使用 `-once` 选项会在一次登录/退出会话之后退出服务器。如果未添加 `-once` 选项，"登录"屏幕会在 Solaris CDE 注销后再次出现。

▼ 使用第一个可用的主机进行登录

- **键入下面的命令并按回车键：**

```
/usr/openwin/bin/X -broadcast
```

该命令会启动 Xserver。在本地子网中，它广播一条 XDM (X 显示器管理器) 登录服务的请求。如果子网中有任何系统正在运行 Solaris CDE 登录管理器 (或其他任何基于 XDM 的登录窗口)，那么第一个响应的主机会将它的登录窗口放到您的桌面上。

特殊的 CDE 配置

本节讲述特殊的配置。

定制邮件打印

CDE 邮件程序使用为 `DTMAIL_FILE` 数据类型定义的打印操作来打印文件。它没有使用 OpenWindows 邮件工具指定的打印正文。要更改打印特性，必须修改此"打印"操作。

▼ 修改打印操作

1. 使用您喜欢的编辑器，创建下面的文件：

HomeDirectory`/.dt/types/dtmail.dt`

2. 在此文件中输入下面的行：

```
#
# Override default Print action for mailboxes
#
ACTION Print
{
LABEL           Print
ARG_TYPE        DTMAIL_FILE
TYPE            COMMAND
WINDOW_TYPE     NO_STDIO
EXEC_STRING     sh -c ' \
               dtmailpr -p -f% (File)Arg_1% | mp -m -l | \
               dtlp -u% (File)Arg_1%;'
}
```

注意：dtmailpr 是一个打印过滤器，它删除附件，产生一个无格式文本流。dtlp 是 lp 的标准 CDE 接口。 %(File)Arg_1% 是正在打印的文件。

---

3. **修改 EXEC_STRING，使其包含需要的打印命令。**
4. **重新启动邮件程序。**

将日历转换到新的数据格式

版本 4 可扩展数据格式是一种 CDE 日历支持的新格式。OpenWindows 日历管理器不能读取这种格式。如果需要在 OpenWindows 和 CDE 平台之间来回转换，就不要将日历转换为版本 4 数据格式。不过，可以用 sdtcm_convert 正文来删改日历。

注意：可以通过从日历"帮助"菜单中选择"关于日历"来查看日历的数据版本。

否则，用 sdtcm_convert 正文将日历转换为版本 4 数据格式。

有关此文件的详细信息，请参阅 sdtcm_convert(1) 手册页。

注意：如果安装的只是“最终用户 CDE 软件包”，就没有安装手册页。若要查看手册页，需要运行 pkgadd 公用程序来安装它。

从网络上添加 AnswerBook 软件包

安装 Solaris CDE 时，安装菜单提供安装 AnswerBook CDE 软件包的选项。如果选择“是”，AnswerBook 软件包会被安装到 /usr/dt/share/answerbooks/ *language* 目录下 (*language* 是具体的 AnswerBook 译本的本地名称)。

因为 AnswerBook 软件包需要 109 兆字节的硬盘空间，所以您可能不想安装这个软件包。不过，可以使用以下两个选项：

- 将 AnswerBook 软件包安装 (或从 /net/. . . 链接) 到 /usr/dt/share/answerbooks/ *language*。
- 添加 AB_CARDCATALOG 环境变量到 *HomeDirectory* /.dtprofile 中。 例如，如果网络上有输出的版本，请添加下面的行到 *HomeDirectory* /.dtprofile：

```
export AB_CARDCATALOG=/net/hostname /usr/dt/share/answerbooks \
    /language/ab_cardcatalog
```

从 CDE 桌面之外设置 CDE

Solaris CDE 应用程序所使用的环境变量可以在 CDE 桌面以外设置。例如，您可能想登录到远程工作站上，并将 CDE 应用程序显示到本地工作站上。请使用 CDE 公用程序 dtsearchpath 来设置各种 CDE shell 环境变量。

对 Bourne 和 Korn shell，请键入下面的命令：

```
eval '/usr/dt/bin/dtsearchpath'
```

对 C shell，请键入下面的命令：

```
eval '/usr/dt/bin/dtsearchpath -c'
```

可以通过设置 DISPLAY 为原来的本地工作站来遵从此命令，然后远程运行 CDE 应用程序。结果显示在本地工作站。

注意：在本例中，Bourne 和 Korn shell 的 dtsearchpath 语法是在所有运行 CDE 的平台上都可用的 CDE 样例执行程序的一部分。但是，C shell (-c) 选项只在 Sun 平台上可用。

桌面环境文件

CDE 桌面为 OpenWindows DeskSet™ 应用程序提供了运行时间的环境。不必对 CDE 设置做任何特殊的操作，就可以获得这个实时环境。此环境设置的一部分是由下面的文件提供的：

```
/usr/dt/config/Xsession.d/0015.sun.env
```

如果对于特定的工作站，需要对此环境设置进行添加或从中删减，此文件既可在原处编辑，也可以首先复制到下面的文件位置：

```
/etc/dt/config/Xsession.d/0015.sun.env
```

增加设置的一个实例是，被 OpenWindows DeskSet 应用程序使用旧的 OpenWindows 虚拟键盘的启动。 因为在 CDE 桌面上，大多数应用程序 (和主要的 Sun 用户) 并不使用这个虚拟键盘公用程序，所以它被放置在缺省的启动序列之外，以改进整个桌面的启动性能。

请参阅 `0015.sun.env` 文件本身的注释，以获取更多有关可选的 OpenWindows 虚拟键盘 (`vkdb`) 启动的信息。

在 Apple Macintosh 应用程序环境下使用软盘和 CD 媒体

如果已经安装 Apple Macintosh 应用程序环境 (MAE) 1.0 版本，并打算通过 OpenWindows 文件管理器应用程序来使用可装卸的媒体 (软盘或 CD)，则必须编辑 `/etc/rmmount.conf` 文件。这可以保证 MAE 正常运行文件管理器的可装卸媒体。

▼ 更改 /etc/rmmount.conf 文件

1. **键入 `su`，以成为超级用户，然后键入口令。**

2. **进入 `/etc` 目录。键入下面的命令并按回车键：**

```
# cd /etc
```

3. **用您喜欢的编辑器打开 `/etc/rmmount.conf` 文件。**

4. **将 `#Actions` 下面的这行移到列表的末尾：**

```
action floppy action_macfs.so
```

例如：

```
# more rmmount.conf

# @(#)rmmount.conf 1.2     92/09/23 SMI

#

# 可装卸媒体安装器配置文件
```

```
#
# 文件系统标识
ident hsfs ident_hsfs.so cdrom
ident ufs ident_ufs.so cdrom floppy
ident pcfs ident_pcfs.so floppy
ident macfs ident_macfs.so floppy
# 操作
action cdrom action_filemgr.so
action floppy action_filemgr.so
action floppy action_macfs.so
```

5. **保存并退出。**

   安装 MAE 后，它将 `macfs action` 行放入 `rmmount.conf` 文件中。卸载 MAE 后，这一行会被删除。

   这一更改使您可以在 OpenWindows 或 CDE 文件管理器应用程序中使用可移动的媒介，同时在 MAE 中仍旧使用 Macintosh 软盘。不过，这也使得 MAE 不能读取未格式化的、难识别的、或 DOS 格式的软盘。使用这个设置过程时，所有这些软盘都会转到 OpenWindows 文件管理器应用程序中。

   如果要允许 MAE 访问不可读的、未格式化的或 DOS 格式的软盘，就必须将 `action floppy action action_macfs.so`行放在 `action floppy action _filemgr.so` 行的前面。

第 7 章

配置在网络中的桌面

桌面专门用于高度网络化的环境。

- 第 109 页"联网桌面概述"
- 第 115 页"配置桌面联网的一般步骤"
- 第 115 页"为桌面配置基本操作系统联网"
- 第 118 页"配置桌面客户和服务器"
- 第 122 页"管理应用程序服务"

桌面的体系结构使系统管理员能够在整个网络中来分布计算资源。其中包括：

- 应用程序
- 应用程序的数据文件
- 桌面会话服务 (诸如"登录管理器"和"文件管理器"之类的桌面应用程序)
- 帮助服务 (帮助数据文件可以放置在中心帮助服务器上)

联网桌面概述

操作系统提供了多种联网服务，其中包括分布式文件系统和远程执行。X 服务器增强了联网能力，其中包括对远程显示器及安全服务的访问。

桌面在这些联网特性之上提供了一个用户界面层。此界面及其下的体系结构的目的在于使网络化系统具有下述特点：

- 便于使用。用户不必担心应用程序和数据究竟在网络中的什么位置，可以直接运行应用程序并且访问数据文件。
- 便于管理。桌面提供了应用程序集成工具和联网的查找路径，它们使系统更容易定位远程数据和应用程序。此外，桌面的文件名映射进程，使得管理包含多服务器的复杂网络更加方便。

■ 性能灵活。虽然桌面的管理特性是为某些特定的通用网络环境而设计的，但该桌面可以适应许多其他定制的网络配置。

联网的桌面服务的类型

联网使用户可以访问分布在其他系统中的各种计算服务，例如：

- 桌面会话及其应用程序—例如，“工作区管理器”和“文件管理器”
- 其他应用程序
- 数据文件

联网术语用**服务器**一词来描述向一个或多个其他系统提供计算服务的系统。当一个系统从服务器获得服务的时候，它就叫做此服务器的**客户机**。

在一套复杂的网络系统中，一个系统可以使用位于整个网络中的若干服务器所提供的服务。此外，一个系统可以既是某一类型的服务器 (例如，会话服务器)，同时又是 (例如，应用程序服务器的) 客户机。

典型的网络情况

从桌面的角度来看，一套典型的网络配置可以包含下面的主要组件的某种组合：

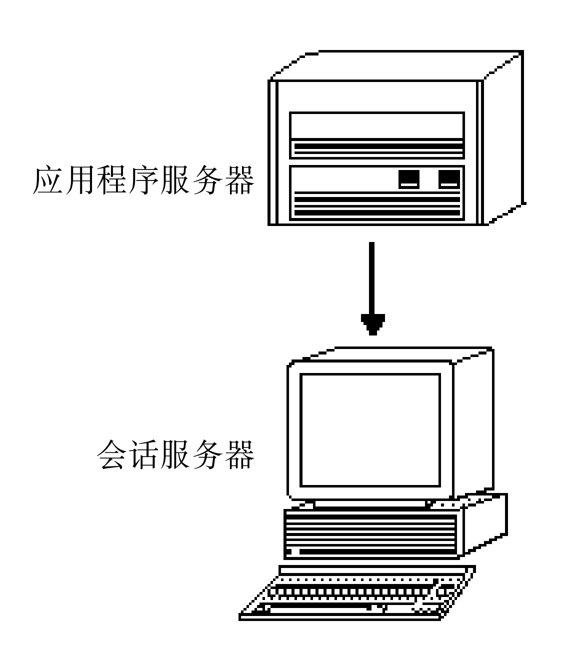
显示器—X 服务器的运行位置

登录/对话服务器—它们运行桌面应用程序 (登录管理器、工作区管理器等。)

应用程序服务器—它们运行其他应用程序

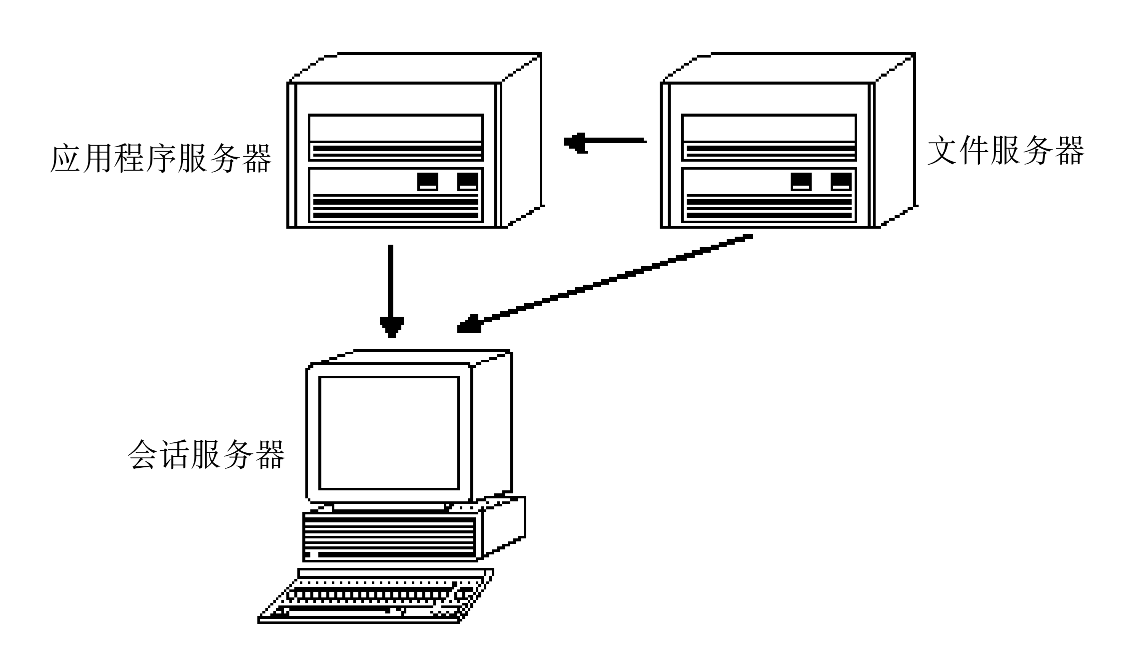
文件服务器—应用程序使用的数据所在的位置

最为通用的网络配置之一包括一些访问应用程序管理器的系统。图形 7–1举例说明了使用应用程序服务器的工作站。此 X 服务器和桌面会话是在该工作站上运行的。



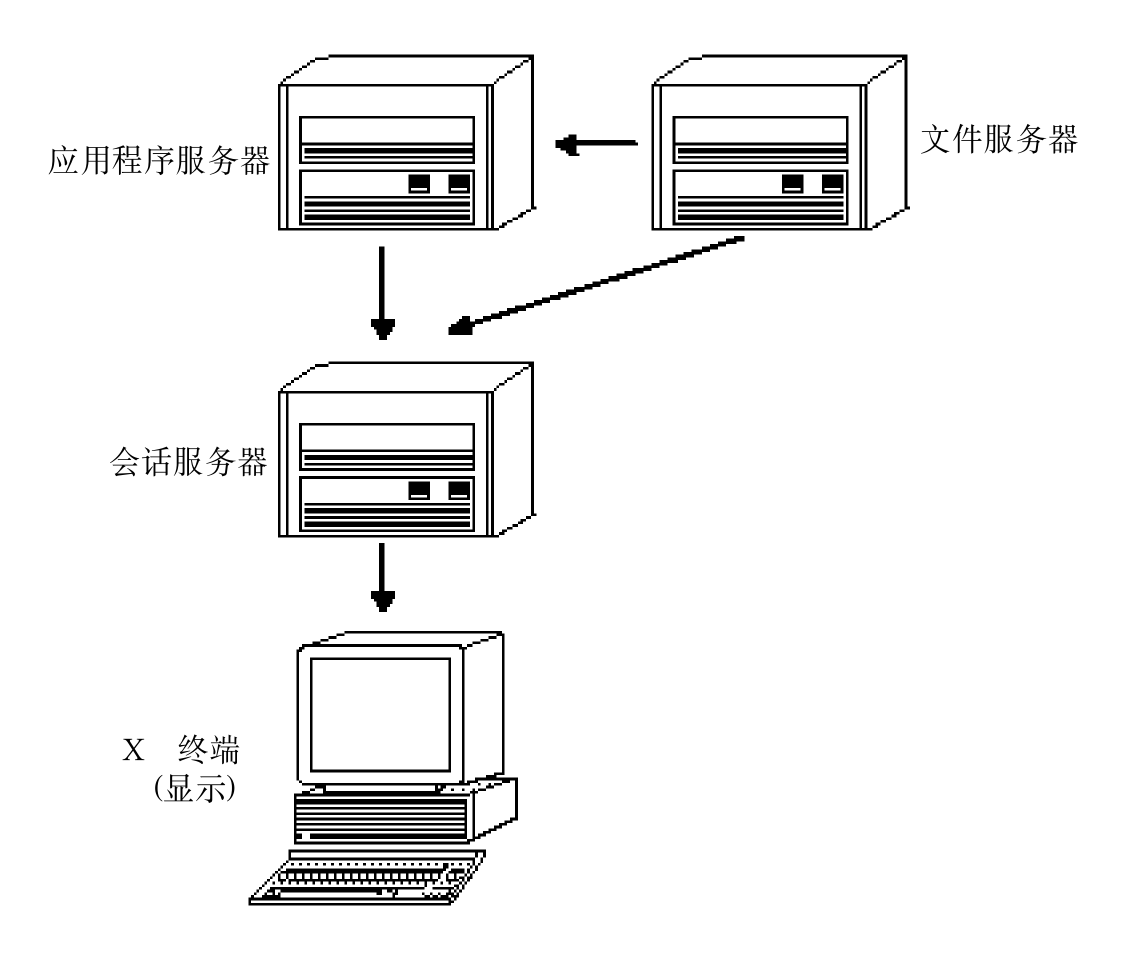
图形 7–1 应用程序服务器为桌面会话提供服务

网络也经常利用文件服务器来存储大量数据。这些数据可能被应用程序服务器上的应用程序使用，或者由桌面应用程序 (例如，“文件管理器”需要访问数据文件，以便将其在“文件管理器”窗口中显示出来) 使用。



图形 7–2 文件服务器为应用程序服务器和会话服务器提供数据

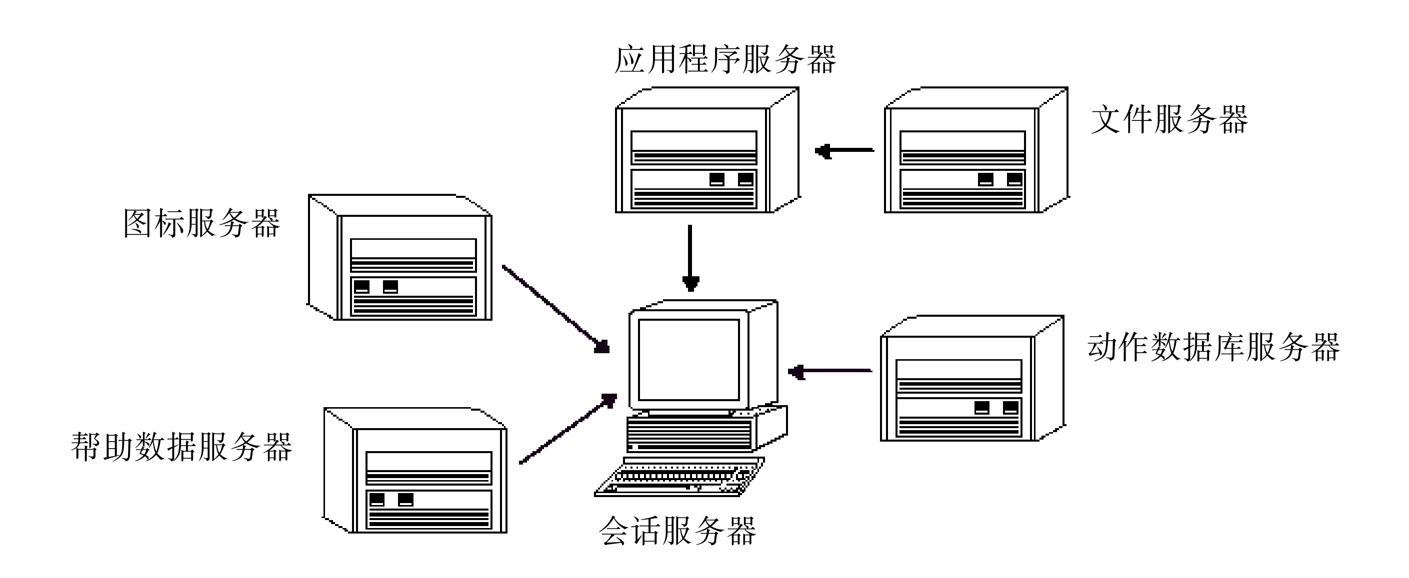
X 终端运行 X 服务器，并从另一系统中获得桌面会话服务。



图形 7–3 X 终端从会话服务器中获得会话服务

其他联网情况

桌面很灵活，可以支持更多复杂的网络配置。除了文件服务器之外， 这通常还包括向应用程序服务器提供各种可用的服务。



图形 7–4 桌面应用程序服务器所需的服务可以是分布式的

总结—服务器的类型

显示器—运行 X 服务器的系统。

登录和会话服务器—运行桌面会话 (如"登录服务器"、"会话服务器"和"窗口服务器"等等) 的系统。

应用程序服务器—运行应用程序的系统。也称为**执行主机**。

文件服务器—存储应用程序数据文件的系统。

帮助服务器—存储帮助文件的系统。

(操作)数据库服务器—存储含有操作和数据类型定义的文件系统。

图标服务器—存储图标文件的系统。

网络还可能包括其他一些服务器，如口令服务器、邮件服务器、视频服务器等等。

配置桌面联网的一般步骤

桌面联网配置一般有三步：

1. **配置基本操作系统的网络服务。**

   这些是由桌面所依赖的操作系统所提供的联网服务。请参阅第 115 页"为桌面配置基本操作系统联网"。

2. **安装并配置桌面联网软件和服务。**

   无论正在建立的客户机系统和服务器系统是什么类型，这些服务都是桌面所必需的。请参阅第 118 页"配置桌面客户和服务器"。

3. **配置服务器和客户机的具体类型。**

   例如，配置一个应用程序服务器就与配置文件服务器的步骤不同，请参阅第 122 页"管理应用程序服务"。

为桌面配置基本操作系统联网

桌面要求下面的基本网络配置：

- 用户必须在会话服务器和为会话服务器提供桌面服务的各个系统上都有登录帐号。在所有的客户和服务器系统中，用户必须有相同的用户 ID 和组 ID 。
- 系统必须能够访问远程文件系统，这些文件系统包含了会话和其他应用程序所使用的数据。
- `lp` 打印假脱机程序必须配置为访问远程打印机。
- 必须配置 `sendmail` 以使用电子邮件服务。
- 必须设置 X 授权。

为用户提供登录帐号

本节描述桌面联网所需要的登录帐号。

提供登录帐号

用户必须有登录帐号在：

- 所有为桌面提供服务的系统，包含应用程序服务器、文件服务器和提供联网打印机的系统。

- 用户可能访问的所有会话服务器。通常，会话服务器是和 X 终端一起使用。

提供一致的用户和组 ID

UNIX 用户是由登录名和数字式用户 ID (UID)来标识的。在桌面网络中，用户在所有的客户和服务器系统中应有相同的登录名和 UID 。

通常指定 UNIX 用户为一个或多个登录组的成员。每组均有其组名和数字组 ID (GID)。在桌面网络中，所有的系统都应有一致的组名和组 ID。

有关详细信息，请参阅 id(1) 或 id(1M) 的手册页。

配置分布式文件系统的访问

桌面利用 NFS 文件系统来进行系统间的文件共享。必须在网络中标识所有包含共享文件的文件系统，并确保它们已正确地安装在所有相应的系统中。

一般地，必须提供下面的远程文件访问：

- 所有桌面客户和服务器系统必须共享用户的起始目录。这是必要的，因为：
  - 用户的起始目录中包含数据文件，这些数据文件是远程系统中的应用程序所必须访问的。例如，使用数据文件的应用程序经常利用用户的起始目录作为缺省的数据文件位置。
  - 用户的起始目录是缺省的 `dtspcd` 认证目录。有关 `dtspcd` 的详细信息，请参阅第 120 页"配置子进程控制守护程序"。
- 如果用户需要访问不在起始目录中的文件，则这些数据文件必须为所有对此文件进行操作的的桌面客户机和服务器系统所共享。
- 桌面安装和配置目录 (`/usr/dt` 和 `/etc/dt`) 必须为所有桌面客户机和服务器系统所共享，这样，用户所有的应用程序就可以访问同样的桌面配置文件。

提供联网的起始目录

若用户只使用一个起始目录，且所有客户和服务器系统均共享此目录， 则此桌面网络系统的工作效率最高。

联网的起始目录可以使得用户使用网络中不同的系统 而不失去个人的定制和配置。这是因为用户个人的定制和用以恢复以前会话的信息被存储在用户起始目录的子目录中。

下面的应用还需要一个通用的起始目录：

- 缺省的 X 授权机制 (请参阅第 117 页"配置 X 授权")。
- 桌面子进程控制守护进程。此守护程序参与到启动远程应用程序的过程之中，而且必须能写到用户的起始目录中。

文件名一致性

应该配置网络，以便用户能够访问存在于所有系统上使用相同的名称的数据文件。这称作提供 **文件名一致性**，通常是通过创建适当的符号链接来完成的。例如，可通过创建符号链接到实际目录安装位置的方法来配置网络中所有系统，使每个用户的起始目录为 `/users/`*login_name* 。

配置对远程打印机的访问

桌面利用 `lp` 打印假脱机程序来访问本地或远程的打印机。有关配置 `lp` 的详细信息，请参阅 lpadmin(1M) 的手册页。

在试图利用桌面图形界面进行打印之前，应首先测试是否能利用 `lp` 命令正确地打印到所有打印机。

确保使用一致的打印设备名。例如，如果某个特定的打印机在与之直接相连的系统中的设备名为 `Postscript1`，则所有其他远程访问此打印机的系统也都应使用 `Postscript1` 的名称来访问。

配置电子邮件

桌面邮件程序利用 `sendmail` 在系统间传递邮件。有关如何配置电子邮件连接的详细信息，请参阅 sendmail(1M) 的手册页。

在试图从桌面发送或接收邮件之前，应当测试一下是否可以用 `mailx` 命令来正确地发送和接收邮件。

配置 X 授权

桌面利用缺省的 X 机制来为访问本地显示器的远程应用程序 (X 客户) 授权。配置的最简便的方法是为每个用户提供联网的起始目录。这就保证了能满足下面的要求：

- 用户必须对文件 *HomeDirectory*`/.Xauthority` 有读写许可。
- 在应用程序服务器上的文件 `.Xauthority` 中必须包含将运行应用程序的显示器 “magic cookie”。

  有关详细信息，请参阅 X(1) 或 xauth(1)的手册页。

配置桌面客户和服务器

本节讲述桌面特有的网络配置要求 —也就是说，这些能力是由桌面提供的， 而不是由基本操作系统所提供的。

本节分为两部分：

- 配置登录和会话服务。
- 配置由应用程序及其其数据所要求的服务。这包括应用程序、数据库、图标、文件及帮助服务器和客户。

配置登录和会话服务

登录/会话服务器是个给显示和 X 服务器提供桌面服务 ("登录管理器"、"会话管理器"、"文件管理器"、"窗口管理器"等) 的系统。

一般地，会话服务器为 X 终端提供服务。但是，可以设置网络配置，将会话服务集中在 X 终端和工作站均可访问的一个或多个服务器上。

"登录管理器"是负责为其他显示器提供登录服务的桌面组件。用户一旦登录，就为用户启动 "会话管理器"。

有关配置登录/会话服务器和 X 终端的信息，请参阅第 23 页"在网络显示器上显示登录屏幕"。

配置其他与应用程序相关的服务

本节包括对通用桌面联网的要求：

- 应用程序服务器
- 数据库服务器
- 图标服务器
- 帮助服务器

▼ 配置桌面客户和服务器

1. **提供桌面所需的操作系统的网络配置。**

   请参阅 第 115 页"为桌面配置基本操作系统联网"。

2. **安装桌面，或最小文件集。**

   必须安装：

- 整个通用桌面环境运行时文件集
- **或**，下面的文件集中的文件：CDE-MIN 和 CDE-TT

注意：安装和文件集可能会因供应商的不同而有差异。

---

3. **为 ToolTalk 文件名数据库服务器守护程序 `rpc.ttdbserverer` 配置系统。**

   安装桌面时，这个操作会自动进行。有关详细信息，请参阅第 121 页"配置 ToolTalk 数据库服务器"。

4. **安装和配置子进程控制守护程序 (`dtspcd`)。**

   安装桌面时，这个操作会自动进行。有关详细信息，请参阅第 120 页"配置子进程控制守护程序"。

5. **安装所需的所有远程数据。**

   当数据不在使用此数据的应用程序所在的系统上的时候， 数据被认为是"远程"的。

   例如：

   - 如果应用程序使用了位于文件服务器上的数据，则它必须安装这些文件。
   - 如果"文件管理器"图标位于图标服务器上，则会话服务器必须装配这些文件。
   - 如果网络使用帮助服务器来存放桌面帮助文件，则会话服务器和所有应用程序服务器必须安装这些帮助数据。

     有关安装点的详细信息，请参见下一节第 119 页"为远程文件系统配置安装点"。

为远程文件系统配置安装点

当桌面将文件名从一个系统传输到另一系统时，它必须将这些文件名变成或**映射**成对目标系统有意义的文件名。由于在不同的系统中同一文件有可能被装配在不同的位置，所以必须用不同的文件名来访问它们，因此这种映射是必需的。例如系统 `sysA` 中的文件 `/projects/big` 可能在 `sysB` 中作为 `/net/sysA/projects/big` 来访问。

文件名映射的要求

为了正确地执行此文件名映射操作，下列之一必须为真：

- `mount` 命令用来静态安装文件系统。这些静态装配的类型一般配置在诸如 `/etc/checklist`、 `/etc/mnttab`、或 `/etc/filesystems` 之类的文件中。

  为了使文件名映射能在系统之间正确地工作，文件系统的安装必须使用一致的主机名。如果主机有几个名称 (例如，别名，或者此主机有不止一个局域网地址，这些地址的名称各不相同) ，必须在所有的安装中使用相同的名称和相同的名称形式。
- **或者**，自动装载程序被用来装载文件系统，装配在缺省的 `/net` 装载点。

- **或者**，自动装载程序用来将文件系统装载在 /net 之外的某个位置，而 DTMOUNTPOINT 环境变量设置为指示此装载点。请参见下一节第 120 页"为 DTMOUNTPOINT 设置值"。

  有关自动装载程序的详细信息，请参见 automount(1M) 的手册页。

▼ 为 DTMOUNTPOINT 设置值

如果下列两个条件均为真，则必须设置 DTMOUNTPOINT 环境变量：

- 自动装载程序用来装载文件系统。
- **而且**，远程文件系统装载在除 /net 之外的某个位置。

  必须为进程设置 DTMOUNTPOINT，包括：

- 在用户登录后会自动启动的用户桌面进程，例如"工作区管理器"(dtwm) 和"文件管理器"(dtfile)
- 由 inetd 之类的机制启动的诸如 rpc.ttdbserver 和 dtspcd 之类的系统进程
- 由本地或远程系统所启动的应用程序。
- 由用户从 shell 命令行启动的应用程序

  为下列的进程设置 DTMOUNTPOINT 变量：

1. **编辑文件** /etc/inetd.conf**：**

   a. **找到** dtspcd **项并添加：**

   ```
   -mount_point mount_point
   ```

   b. **找到** rpc.ttdbserver **项并添加：**

   ```
   -m mount_point
   ```

   例如，如果自动装载程序正用于安装点 /nfs，则在 /etc/inetd.conf 中的项为：

   ```
     dtspc stream tcp nowait root /usr/dt/bin/dtspcd \
   /usr/dt/bin/dtspcd -mount_point /nfs
   rpc stream tcp wait root /usr/dt/bin/rpc.ttdbserver \
   100083 1 rpc.ttdbserver -m /nfs
   ```

2. **在重新读取** /etc/inetd.conf **的系统上执行此进程。有关详细信息，请参见 inetd (1M) 的手册页。**

3. **使用由用户登录中继承来的值设置 DTMOUNTPOINT 。**

   这可通过在 /etc/dt/config/Xsession.d 中设置变量的方法来完成。有关设置环境变量的详细信息，请参阅第 44 页"设置环境变量"。

配置子进程控制守护程序

桌面的子进程控制 (SPC) 服务提供客户/服务器命令的执行。

桌面子进程控制守护程序 (dtspcd) 用于桌面启动远程应用程序。它是个接收远程客户执行命令的请求的 inet 守护程序。有关如何配置 inet 守护进程的详细信息，请参阅 inetd.conf(1M) 的手册页。

桌面的操作调用库使用 SPC 服务来调用远程操作。

▼ 配置 dtspcd

- **请确定** dtspc **是否正确注册到** /etc/services **和** /etc/inetd.conf **中。**

请参阅 dtspcd(1M) 的手册页。

SPC 安全

对于子进程控制服务的认证是在文件系统验证的基础上进行的。dtspcd 必须能访问所有 SPC 客户系统均装载的**认证目录**。

缺省情况下，dtspcd 的验证目录就是用户的起始目录。但可通过设置目录 -/etc/inetd.conf 中的 auth_dir 选项来配置 dtspcd，以使用不同的目录。有关详细信息，请参阅 dtspcd(1M)的手册页。

由于 SPC 认证是在文件系统验证的基础上进行的，所以 SPC 服务仅仅与分布式文件系统的安全性一样。如果正在一个您不信任其分布式文件系统的网络中使用桌面的话，您可能想禁用 dtspcd。要禁用 dtspcd，需注释出 /etc/services 中的 dtspc 项。

为远程执行配置环境变量

当桌面利用操作来启动远程系统上的应用程序的时候，用户的环境变量被复制到远程系统上，并被放入相应的应用程序的环境中。

缺省情况下，更改某些环境变量，然后把它们复制到远程系统中。在环境变量放入应用程序的环境之前，可通过配置操作调用组件和桌面的子进程控制服务来执行附加环境变量的处理。

有关缺省的配置和如何对其进行修改的详细信息，请参阅 dtactionfile(4) 和 dtspcdenv (4) 的手册页。

配置 ToolTalk 数据库服务器

ToolTalk 组件之一是 ToolTalk 数据库服务器， /usr/dt/bin/rpc.ttdbserver。

ToolTalk 消息服务使用 ToolTalk 数据库服务器，此服务器还用于文件名映射。它通常在安装桌面时注册到 /etc/inetd.conf 中，因此不需要附加的配置工作。

有关 ToolTalk 数据库服务器及其配置选项的详细信息，请参阅 rpc.ttdbserver(1M) 的手册页。

配置 ToolTalk 消息服务器

ToolTalk 消息服务器是 `ttsession`。缺省情况下，它不需任何配置；在登录期间，它由 `X` 会话正文启动。

有关 ToolTalk 消息服务器及其配置选项的详细信息，请参见 ttsession 的手册页。

配置日历守护程序

"日历"应用程序的组件之一是"日历守护程序"`rpc.cmsd`。它通常在安装桌面时注册到 `/etc/inetd.conf` 中，因此不需要附加的配置工作。

有关日历守护程序及其配置选项的详细信息，请参阅 rpc.cmsd(1) 的手册页。

管理应用程序服务

本节讲述下列各项的具体配置要求：

- 应用程序服务器及其客户户
- 提供特殊服务的桌面服务器—数据库服务器、图标服务器和帮助服务器

  还讲述对两个联网的应用程序的特殊配置的联网要求：
- 远程执行的主机
- 在文件系统安装之间运行的应用程序。

查找路径环境变量

桌面使用一系列环境变量来指定用来查找应用程序的桌面配置文件的查找路径，这些配置文件包括操作和数据类型数据库、帮助文件和图标文件。

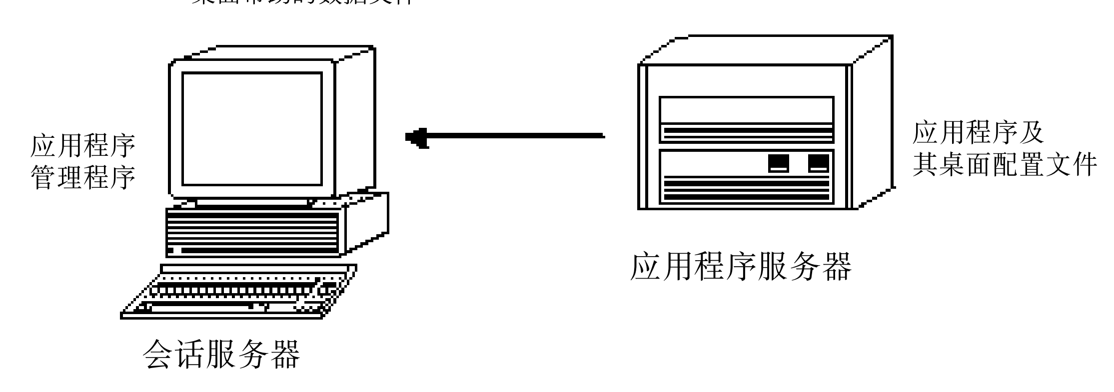
有关如何使用查找路径环境变量的详细信息，请参阅第 136 页"桌面查找路径及其环境变量" 或 dtenvvar(5) 手册页。

配置应用程序服务器及其客户机

在标准的应用程序服务器的配置中，应用程序服务器包含与应用程序并联的所有二进制文件和配置文件，包括：

- 应用程序的可执行文件
- 标准的应用程序配置文件，诸如 app-defaults、消息目录和此应用程序所需的共享库之类。

- 桌面配置文件：
  - 操作和数据类型的定义文件
  - 图标图像文件
  - 桌面帮助的数据文件



图形 7–5 标准应用程序服务器的配置

▼ 配置应用程序服务器

1. **提供桌面所需的操作系统的网络配置。**

   请参见第 115 页"为桌面配置基本操作系统联网"。

2. **提供服务器所需的通用桌面配置。**

   请参阅第 118 页"配置桌面客户和服务器"。

3. **安装应用程序。**

4. **如果应用程序不能自动注册，则必须执行注册过程。**

   请参见 第 5 章。

▼ 配置应用程序服务器客户机

1. **提供桌面所需的操作系统的网络配置。**

   请参见第 115 页"为桌面配置基本操作系统联网"。

2. **提供客户机所需的通用桌面配置。**

   请参见第 118 页"配置桌面客户和服务器"。

3. **在系统范围内或个人范围内将应用程序服务器添加至应用程序查找路径中：**

   全系统—设置并导出 `/etc/dt/config/Xsession.d/0010.dtpaths` 中的 `DTSPSYSAPPHOSTS` 变量

个人—设置并导出 HomeDirectory /.dtprofile 中的 DTSPUSERAPPHOSTS 变量

例如， 如下所示的在 /etc/dt/config/Xsession.d/0010.dtpaths 中的行将在应用程序查找路径中添加带主机名的系统 SysAAA 和 SysBBB：

```
export DTSPSYSAPPHOSTS=SysAAA:，SysBBB:
```

有关设置应用程序查找路径的详细信息，请参见：

- 第 137 页"应用程序查找路径"
- 第 136 页"设置查找路径的值"

配置数据库、图标和帮助服务

通常，与应用程序相关的操作和数据类型定义、图标和帮助数据文件是与应用程序安装在同一系统上的。

例如，可考虑以下典型的帮助数据文件的配置：

- "文件管理器"的帮助文件通常是在会话服务器之上的。因为帮助的查找路径会自动查找会话服务器上的正确位置，所以桌面能找到这些帮助文件。
- 其他应用程序的帮助文件通常与其相应的应用程序位于同一应用程序服务器上。因为修改应用程序的"查找路径"过程会自动修改帮助查找路径，所以会话服务器能找到这些帮助文件。

  有时可能需要使数据库 (操作和数据类型)、帮助或图标数据放置在网络中的其他位置。例如，如果网络中使用了多个会话服务器，就有可能要创建帮助服务器，在此帮助服务器中存储桌面应用程序 (文件管理器、式样管理器等等) 的所有帮助数据文件。由于帮助文件不需在每个会话服务器上均复制一份，因而节省了磁盘空间。

▼ 创建数据库、帮助或图标服务器

1. **提供桌面所需的操作系统的网络配置。**

   请参见 第 115 页"为桌面配置基本操作系统联网"。

2. **提供客户机所需的通用桌面配置。**

   请参见 第 118 页"配置桌面客户和服务器"。

3. **安装数据库、帮助或图标文件。**

   这些文件可以处于系统的任何地方。但是，当为系统指定了应用程序服务器后，会自动查找下列目录，所以使用下列位置会更方便。

   - 数据库文件：/etc/dt/appconfig/types/*language*
   - 帮助文件： /etc/dt/appconfig/help/*language*
   - 图标文件： /etc/dt/appconfig/icons/*language*

     如果正在设置数据库服务器，则必须写一些操作来指定它们的 EXEC_STRING命令将在何处运行。请参见第 125 页"指定远程执行主机"。

▼ 配置会话服务器以找到数据库、图标或帮助服务器

1. **提供桌面所需的操作系统的网络配置。**

   请参见 第 115 页"为桌面配置基本操作系统联网"。

2. **提供客户机所需的通用桌面配置。**

   请参见 第 118 页"配置桌面客户和服务器"。

3. **将数据库、图标或帮助服务器添加到适当的查找路径中。**

   - 如果将数据文件放置在 第 124 页"创建数据库、帮助或图标服务器"的 步骤 3中指定的位置，您可以修改应用程序查找路径。
   - 如果将数据文件放置在其他位置，就必须修改特定的查找路径。

     例如，如果将帮助文件放置在系统 SysCCC 的 目录 `/etc/dt/help` 下，则必须将下行添加到 /etc/dt/config/Xsession.d/0010.dtpaths 中：

     ```
     export DTSPSYSHELP=/net/SysCCC/etc/dt/help
     ```

     有关设置查找路径的详细信息，请参见：

   - 第 140 页"数据库 (操作/数据类型) 查找路径"
   - 第 142 页"图标查找路径"
   - 第 144 页"帮助查找路径"
   - 第 136 页"设置查找路径的值"

特殊的、联网应用程序的配置

本节讲述如何配置系统以运行应用程序：

- 不是在包含操作的系统上—而是在远程执行主机上
- 在本地跨文件系统安装

指定远程执行主机

在典型的应用程序服务器的配置中，操作定义与应用程序可执行文件同在一个系统中。但也可以写一些操作来执行其他系统上的命令。在此配置下，包含应用程序的系统被称为**执行主机**。

操作定义可以定位在会话服务器上，也可以在为会话服务器提供操作和数据类型服务的系统上—被称为**数据库服务器** 或**数据库主机**。

操作的定义使用 `EXEC_HOST` 字段来指定其 `EXEC_STRING` 命令应在何处运行。例如，下面的操作定义指定了 `xload` 客户机程序运行在主机名为 `SysDDD`的系统上：

```
Action XloadSysDDD
  {
   TYPE         COMMAND
```

```
  EXEC_HOST         SysDDD
  EXEC_STRING         /usr/bin/X11/xload -label SysDDD
 }
```

如果 EXEC_HOST 字段中指定了不止一个主机名，则桌面按顺序在每个主机上执行 EXEC_STRING，直至找到能执行此操作的主机。例如，下面的 EXEC_HOST 字段指定了操作应首先在 SysDDD 系统上尝试运行 EXEC_STRING，如此运行失败，则尝试 SysEEE。

```
   EXEC_HOST                SysDDD，SYSEEE
```

如果不为某个操作而设置 EXEC_HOST 字段不是为操作而设置，便使用其缺省值 %DatabaseHost%。值 %DatabaseHost% 是从数据库的查找路径中得到的。

例如，假设数据库的查找路径因为向 /etc/dt/config/Xsession.d/0010.dtpaths 中添加下行内容而更改：

```
DTSPSYSDATABASEHOSTS=SysAAA：，/net/SysBBB/etc/dt/appconfig/types/C
```

SysAAA 是由主机限定语法—SysAAA：指定的。发现使用查找路径中的此元素的某个操作定义会将数据库主机设置为 SysAAA。但是，发现使用查找路径中的 /net/SysBBB ... 这一部分的操作会将数据库主机设置为本地系统，因为语法不包括主机限定符。

▼ 配置远程执行主机

1. **提供桌面所需的操作系统的网络配置。**

   请参见第 115 页“为桌面配置基本操作系统联网”。

2. **提供服务器所需的通用桌面配置。**

   请参见第 118 页“配置桌面客户和服务器”。

3. **确保应用程序已经为本地执行而正确安装和配置。**

▼ 配置包含操作定义的系统

1. **提供桌面所需的操作系统的网络配置。**

   请参见第 115 页“为桌面配置基本操作系统联网”。

2. **提供服务器所需的通用桌面配置。**

   请参见第 118 页“配置桌面客户和服务器”。

3. **创建并安装操作定义和应用程序组。**

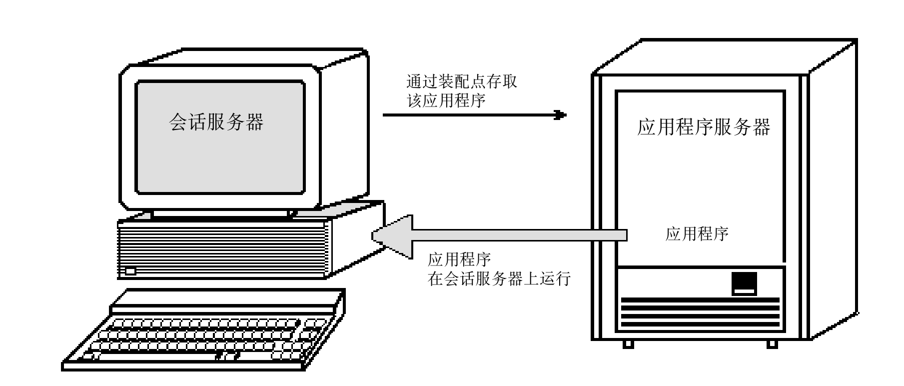
   请参见第 188 页“创建在远程系统上运行应用程序的操作” 和 第 59 页“创建和管理通用应用程序组”。

▼ 配置会话服务器

1. **提供桌面所需的操作系统的网络配置。**
   请参见第 115 页“为桌面配置基本操作系统联网”。
2. **提供客户机所需的通用桌面配置。**
   请参见第 118 页“配置桌面客户和服务器”。
3. **修改操作查找路径，使其包含数据库主机。**
   请参见第 140 页“数据库 (操作/数据类型) 查找路径”。
4. **修改应用程序查找路径，使其包含执行主机。**
   请参见第 137 页“应用程序查找路径”。

在本地运行应用程序

标准的应用程序服务器配置是在应用程序服务器上运行应用程序。但有时将应用程序安装在远程系统上而在本地的会话服务器上执行是可取的。



图形 7–6 通过安装点执行

▼ 配置应用程序服务器

不需要特殊的配置。

▼ 配置会话服务器

- **修改应用程序查找路径。利用到应用程序的本地绝对路径 。**

例如，可利用下面的变量定义来找到在系统 sysAAA 上注册的某一应用程序：

```
DTSPSYSAPPS=/net/SysAAA/etc/dt/appconfig/appmanager/C
```

会话服务器必须能访问诸如 app-defaults、消息目录和共享库之类的应用程序配置文件。

第 8 章

从桌面打印的配置和管理

桌面用户打印文件可以有几种不同的方法。它们分为两大类：从桌面上进行打印和从应用程序中打印。

桌面打印的方法包括：

- 在"文件管理器"中选择文件，然后从"已选择的"菜单中或图标的弹出式菜单中选择"打印"
- 从"文件管理器"中拖动文件到"前面板"的"打印机"控制上或"个人打印机"子面板中
- 从"文件管理器"中拖动文件到"打印管理器"主窗口中的打印机上

  如果要从应用程序中打印，可使用"打印"命令，该命令通常可从应用程序窗口的菜单或其他控件中获得。

增加和删除打印机

本节讲述了从桌面增加和删除打印机的过程。

▼ 向桌面上增加打印机

1. **增加打印机到系统配置中。**

   根据操作系统的系统管理文档中的指示进行操作。

2. **运行命令：**

```
env LANG=language  /usr/dt/bin/dtprintinfo -populate
```

3. **重新启动"打印管理器"，或者在"应用程序管理器"的 Desktop_Tools 应用程序组中双击"重新装入操作"。确认打印机已经出现。**

4. **向用户发送邮件，通知他们重新启动"打印管理器"或运行"重新装入操作"。**

   "打印管理器"在每次调用时都会读取系统打印机配置列表。如果检测到新的打印机，它会自动创建新的桌面打印机操作并为该打印机创建图标。无须做任其他何事情打印机就可以显示在桌面上。

▼ 从桌面上删除打印机

1. **从系统配置中删除打印机。**

   根据操作系统的系统管理文档中的指示进行操作。

2. **重新启动"打印管理器"，或者在"应用程序管理器"的 Desktop_Tools 应用程序组中双击"重新装入操作"。确认打印机已经去除。**

3. **向用户发送邮件，通知他们重新启动"打印管理器"或运行"重新装入操作"。**

   "打印管理器"在每次调用时都会读取系统打印机配置列表。如果它发现打印机已经从列表中删除，它会自动从"打印管理器"和"文件管理器"中删除该打印机的操作及其图标。无须做任何其他事情就可以从桌面上删除打印机。

注意："打印管理器"无法从"前面板"上删除打印机。所以，无论何时从配置中删除打印机，都应该向系统中的所有用户发送邮件，告诉他们从"前面板"中删除该打印机的图标。

修改作业更新间隔

要更改"打印管理器"中显示信息的更新频率，可修改作业更新间隔。在缺省情况下，"打印管理器"每隔三十秒查询一次打印机有关打印作业的信息。使用"设置选项"对话框中的"更新间隔"滑块可更改"打印管理器"查询打印机的频率，而该对话框由选择"视图"菜单中的"设置选项"显示。

打印机图标图像

在增加打印机时，系统自动为增加的打印机指定了缺省的打印机图标。如果要为该打印机指定其他的图标，可将该图标文件放到 /etc/dt/appconfig/icons/*language*目录中，或放到图标查找路径的其他目录中。这样，用户就可以选择该图标替换打印机的缺省图标。

必须创建一整套图标 (大号、中号、小号图标)，否则它们将不会出现在“打印管理器”的图标选择程序中。

有关图标查找路径的详细信息，请参见第 142 页“图标查找路径”。

图标文件名和大小

图标文件命名要求是：

base_name.size.*type*

其中：

size —l (大号)、m (中号)、t (小号)。 有关图标大小的详细内容，请参见第 210 页“图标大小惯例”.

type —pm (彩色像素图)、bm (位图)。

例如，彩色打印机的中号和小号像素图图标的图标文件名可能会是 `ColorPrinter.m.pm`和 `ColorPrinter.t.pm`。

有关创建图标的详细信息，请参见第 14 章。

▼ 全局更改打印机的图标、打印机的标号或打印机的描述

应当一增加打印机在用户使用“打印管理器”修改打印机属性前，就尽快更改全局打印机特性。因为一旦用户用“打印管理器”修改了打印机属性，那么他们将看不到您做的更改。

用期望的信息为图标、打印机标号或描述编辑 /etc/dt/appconfig/types/*language*/*printer_queue_name*.dt 文件：

1. **在 `ICON` 字段，更新 *basename* 为新图标的基本名称。**
2. **在 `LABEL` 字段，更新 *labelname* 为打印机的新标记。**
3. **更新在 `DESCRIPTION` 字段中的文本。**

   这是用于放置打印机位置、打印机型号和打印机联系人的好地方。要增加的内容不只一行，可在行尾处放置 \。例如：

```
DESCRIPTION              This is a PostScript Printer in Building 1 \
                 Room 123. Call 555-5555 for problems.
```

配置缺省打印机

用户执行下面的操作时访问缺省打印机：

- 拉放对象到“前面板”的“打印机”控制上
- 从“文件管理器”中选择目标，然而从“已选择的”菜单中或图标的弹出式菜单中选择“打印”
- 从使用缺省打印机的应用程序中打印

▼ 更改缺省的打印目标

为所有用户更改缺省打印机：

1. **打开 `/etc/dt/config/Xsession.d/0010.dtpaths` 文件。**

   如果 `/etc/dt/config/Xsession.d/0010.dtpaths` 文件不存在，那么从 `/usr/dt/config/Xsession.d/0010.dtpaths` 处复制它。

2. **在 `LPDEST=`*printer* 行中，更新 *printer* 为新的缺省打印机目标。**

   如果这行不存在，那么增加 `LPDEST=`*printer* 行，其中 *printer* 是所希望的缺省打印机的名称。

3. **用户需要注销并重新登录。**

要为单个用户更改缺省打印机，该用户应该：

● **从“个人打印机”子面板中将另一个打印机复制到“前面板”中。**

如果要指定其他打印机作为缺省打印机：

1. **转至起始文件夹，并打开 `.dtprofile` 文件。**
2. **增加或编辑用于设置 `LPDEST` 环境变量数值的行：**

   `LPDEST=`*printer_device;* `export LPDEST`

   如果正在使用 `csh`，语法为：

   `setenv LPDEST` *printer_device*

   例如，下面一行将更改缺省打印机为设备名为 `laser3d` 的打印机。

```
LPDEST=laser3d; export LPDEST
```

如果正在使用 `csh`，语法为：

```
setenv LPDEST laser3d
```

打印概念

由拉放文件到打印机控制上，启动打印请求时，系统进行如下操作：

1. 系统查找拉放对象定义的数据类型数据库。
2. 如果对该数据类型 (由打印操作中的 ARG_TYPE 字段指定) 有唯一的打印操作，那么使用该打印操作；否则，使用缺省打印操作 (dtlp)。例如，如果文件是 **PostScript** ® 文件，那么系统使用“**PostScript** 文件”的“打印” 操作。(该操作在 /usr/dt/appconfig/types/*language* /dt.dt 中定义。)如果对该数据类型使用“创建操作”工具，那么输入的打印命令就是用于打印该数据类型文件的唯一打印操作。
3. 使用标准 **UNIX** lp 打印子系统，将文件发送到打印机。

第 9 章

桌面查找路径

桌面利用查找路径来定位应用程序和与它们相关联的桌面文件。

桌面提供四种查找路径，如表 9–1中所述。

表 9–1 桌面查找路径

查找路径	描述
应用程序	用于定位应用程序。当用户登录时，"应用程序管理器"使用应用程序查找路径来动态地总装其顶层。
数据库	用于指定操作和数据类型定义文件 (`*.dt` 文件) 与"前面板文件" (`*.fp` 文件) 两者的附加位置。
图标	用于指定图标的附加位置。
帮助数据	用于指定桌面帮助数据的附加位置。

查找路径可以包含本地和远程目录。因此，查找路径在桌面的网络体系结构中扮演一个重要的角色。例如，正因为在应用程序查找路径中列出了应用程序服务器，才使系统可以在那个应用程序服务器上查到应用程序。

当查找路径包含远程位置时，必须配置远程文件访问该位置。有关详细信息，请参阅第 116 页"配置分布式文件系统的访问"。

桌面查找路径及其环境变量

在登录时，由桌面公用程序 dtsearchpath 创建桌面查找路径。dtsearchpath 公用程序利用环境变量与内置位置的组合来创建查找路径。

dtsearchpath 读取的环境变量被称作**输入变量**。 这些是由系统管理员和最终用户设置的变量。输入变量使用命名惯例 DTSP*。

在登录运行 dtsearchpath 时，它组合为这些变量指定的值，加入内置位置，并创建**输出变量**的值。对应每个查找路径有一个输出变量。

表 9–2 桌面查找路径环境变量

查找路径为:	输出环境变量	系统范围输入变量	个人输入变量
应用程序	DTAPPSEARCHPATH	DTSPSYSAPPHOSTS	DTSPUSERAPPHOSTS
数据库[1]	DTDATABASESEARCHPATH	DTSPSYSDATABASEHOSTS	DTSPUSERDATABASEHOSTS
图标	XMICONSEARCHPATH，XMICONBMSEARCHPATH	DTSPSYSICON	DTSPUSERICON
帮助数据	DTHELPSEARCHPATH	DTSPSYSHELP	DTSPUSERHELP

[1]Actions, data types, and Front Panel definitions

组件使用输出变量的值。例如，"应用程序管理器"使用应用程序查找路径的值 (DTAPPSEARCHPATH) 来定位应用程序组。

设置查找路径的值

可以在系统或个人范围内修改查找路径。通过设置用于系统范围或个人输入变量的值来完成修改。所作的任何修改将被添加到内置的查找路径位置中。

▼ 查看查找路径的当前值 (输出变量)

● **使用 dtsearchpath 命令来显示查找路径的当前值：**

- 要获得当前 (登录) 用户的值，请键入命令：

```
dtsearchpath -v
```

- 要获得不同用户的值，请键入命令：

```
dtsearchpath -u user
```

查找路径值包含这些变量：

%H—在 DTHELPSEARCHPATH 中使用。帮助的卷名。

%B—在 XMICONSEARCHPATH 中使用。图标文件的基本名称。

%M—在 XMICONSEARCHPATH 中使用的图标文件的大小 (.l、.m、.s、.t).

%L— LANG 环境变量的值。

▼ 对查找路径作个人修改

1. **打开** *HomeDirectory* /.dtprofile **用于编辑。**
2. **增加或编辑定义和导出个人输入变量的一行。**

   例如，下面一行向用户的个人应用程序查找路径增加位置：

   ```
   export DTSPUSERAPPHOSTS=/projects1/editors
   ```

3. **要使更改生效，注销并重新登录。**

▼ 对查找路径作系统范围修改

1. **以超级用户登录。**
2. **如果文件 /**etc/dt/config/Xsession.d/0010.dtpaths **不存在，就通过复制** /usr/dt/config/Xsession.d/0010.dtpaths **来创建它。**
3. **打开 /**etc/dt/config/Xsession.d/0010.paths**，用于编辑。添加或编辑定义和导出系统范围输入变量的一行。**

   例如，下面一行向系统范围的帮助查找路径中增加位置：

   ```
   export DTSPSYSHELP=/applications/helpdata
   ```

4. **通知系统中的所有用户，他们必须注销并重新登录，才能使更改生效。**

应用程序查找路径

应用程序查找路径是主查找路径，桌面用它来定位在本地系统和遍及整个网络的应用程序服务器上的应用程序。

向应用程序查找路径增加位置时，其他查找路径 (数据库、图标和帮助) 就自动地更新，以反映该数据的相应位置；因此，应用程序查找路径为应用程序及其桌面配置文件提供了相对简便的管理方式。请参见第 139 页"应用程序查找路径如何影响数据库、图标和帮助的查找路径"。

缺省的应用程序查找路径

缺省应用程序查找路径包含个人的、系统范围的和内置的位置。缺省的 *language* 是 C。

个人位置— *HomeDirectory* /.dt/appmanager

系统范围位置—/etc/dt/appconfig/appmanager/*language*

内置位置—/usr/dt/appconfig/appmanager/*language*

应用程序查找路径环境变量

应用程序查找路径由内置位置和下面的输入变量组合而成：

DTSPSYSAPPHOSTS—系统范围应用程序查找路径输入变量

DTSPUSERAPPHOSTS—个人应用程序查找路径输入变量

组合查找路径由输出变量 DTAPPSEARCHPATH 来指定。

应用程序查找路径输入变量的语法

变量 DTSPSYSAPPHOSTS 和 DTSPUSERAPPHOSTS 的语法是：

VARIABLE=location [,*location*...]

其中，*location* 具有此语法的位置：

/*path*—指定本地 (会话服务器) 系统上的目录。 使用此语法来添加本地目录。

hostname:—指定在系统 *hostname* 上的系统范围目录 /etc/dt/appconfig/appmanager/*language*。使用此语法来添加应用程序服务器。

hostname:/*path*— 指定在远程系统 *hostname* 上的一个目录。

hostname:—本地系统范围位置。此关键字用来改变本地系统范围位置的优先级。请参阅第 139 页"更改系统范围本地位置的优先级"。

如何组合应用程序查找路径的值

通过组合下面按优先级顺序列出的位置，来创建应用程序查找路径的值 (DTAPPSEARCHPATH)：

- 使用 DTSPUSERAPPHOSTS 变量指定的位置
- 缺省个人位置： *HomeDirectory* /.dt/appmanager
- 缺省位置：/etc/dt/appconfig/appmanager/*language*
- 使用 DTSPSYSAPPHOSTS 变量指定的位置
- /usr/dt/appconfig/appmanager/*language*

  语法：

  hostname：

  扩充以指定在系统 *hostname* 上的 /etc/dt/appconfig/appmanager 目录。

更改系统范围本地位置的优先级

缺省情况下，本地的系统范围位置 (/etc/dt/appconfig/appmanager/*language*) 具有高于远程位置的优先级。因此，本地应用程序组具有高于同名的远程组的优先级。例如，如果本地和远程系统都有"打印机"应用程序组 (/etc/dt/appconfig/appmanager/*language* /Printers)，这时要用本地组。

应用程序查找路径输入变量提供一种语法，用于指定本地系统范围应用程序组的优先级：

```
localhost:
```

例如，假设系统必须访问应用程序服务器 SysA、SysB 和 SysC，且想要在 SysB 上的系统范围应用程序组具有高于同名的任何本地组的优先级。

下面的 DTSPSYSAPPHOSTS 的值创建此性能：

```
DTSPSYSAPPHOSTS=SysB:，localhost:，SysA:，SysC:
```

应用程序查找路径如何影响数据库、图标和帮助的查找路径

对应用程序查找路径的补充会自动将相应的位置添加到数据库、图标和帮助的查找路径。这样，只要设置应用程序查找路径输入变量就能将应用程序服务器添加到查找路径。

例如，如果设置 DTSPSYSAPPHOSTS 如下：

```
export DTSPSYSAPPHOSTS=servera:
```

则查找路径受影响，如表 9–3所示。

表 9–3 受影响的查找路径

查找路径	增加到查找路径的目录
应用程序	servera:/etc/dt/appconfig/appmanager/*language*
数据库	servera:/etc/dt/appconfig/types/ *language*
图标	servera:/etc/dt/appconfig/icons/*language*
帮助	servera:/etc/dt/appconfig/help/*language*

同样地，如果设置 DTSPSYSAPPHOSTS 如下：

```
export DTSPSYSAPPHOSTS=/projects1/apps
```

则查找路径受影响，如表 9–4所示。

表 9–4 受影响的查找路径

查找路径	增加到查找路径的目录
应用程序	/projects1/apps/appmanager/*language*
数据库	/projects1/apps/types/ *language*
图标	/projects1/apps/icons/*language*
帮助	/projects1/apps/help/*language*

数据库 (操作/数据类型) 查找路径

数据库查找路径指导桌面来搜索指定的位置，以便包含下列内容的文件：

- 操作和数据类型定义 (*.dt 文件)
- “前面板”定义 (*.fp 文件)。

  当创建数据库服务器，或为数据库文件添加本地位置时，可能需要修改数据库的查找路径。

缺省的数据库查找路径

缺省数据库查找路径包含个人的、系统范围的和内置的位置。缺省的 *language* 是 C。

个人位置— *HomeDirectory* /.dt/types

系统范围位置—/etc/dt/appconfig/types/*language*

内置位置—/usr/dt/appconfig/types/*language*

应用程序查找路径如何影响数据库查找路径

向应用程序查找路径添加位置时，相应的数据库子目录会自动地增加到数据库查找路径中 (请参见第 139 页"应用程序查找路径如何影响数据库、图标和帮助的查找路径")。

例如，如果应用程序服务器 hosta: 被添加到应用程序查找路径，那么目录 hosta:/etc/dt/appconfig/types/*language* 就自动地添加到数据库查找路径。

数据库查找路径环境变量

数据库查找路径由内置位置和下面的输入变量组合而成：

DTSPSYSDATABASEHOSTS—系统范围数据库查找路径输入变量

DTSPUSERDATABASEHOSTS—个人数据库查找路径输入变量

使用这些输入变量来指定在应用程序查找路径之外的位置。

已组合的数据库查找路径是由输出变量 DTDATABASESEARCHPATH 来指定。

数据库查找路径输入变量的语法

变量 DTSPSYSDATABASEHOSTS 和 DTSPUSERDATABASEHOSTS 的语法是:

VARIABLE=location [,*location*...]

其中，*location* 可以有语法：

/*path*—指定本地 (会话服务器) 系统上的目录。使用此语法来添加本地目录。

hostname：—指定在系统 *hostname* 上的系统范围目录 /etc/dt/appconfig/types/*language*。

hostname:/*path*— 指定远程系统 *hostname* 的目录。

如何组合数据库查找路径

数据库查找路径的值 (DTDATABASESEARCHPATH) 是通过组合下面的按优先级顺序列出的位置来创建的：

- 使用 DTSPUSERDATABASEHOSTS 变量指定的位置

- 由 DTSPUSERAPPHOSTS 变量导出的位置
- 缺省个人位置： *HomeDirectory* /.dt/types
- 缺省位置：/etc/dt/appconfig/types/*language*
- 使用 DTSPSYSDATABASEHOSTS 变量
- 由 DTSPSYSAPPHOSTS 变量导出的位置
- /usr/dt/appconfig/types/*language*

  语法：

  hostname：

  扩充以指定在系统 *hostname* 上的 /etc/dt/appconfig/appmanager 目录。

图标查找路径

图标查找路径指导桌面来搜索指定文件位置，这些文件包含桌面所使用的位图和像素图影象文件。

缺省的图标查找路径

缺省的图标查找路径包含个人的、系统范围的和内置的位置。缺省的 *language* 是 C。

个人位置— *HomeDirectory* /.dt/icons

系统范围位置—/etc/dt/appconfig/icons/*language*

内置位置—/usr/dt/appconfig/icons/*language*

应用程序查找路径如何影响图标查找路径

向应用程序查找路径增加位置时，相应的图标子目录将自动地增加到图标查找路径 (请参见第 139 页"应用程序查找路径如何影响数据库、图标和帮助的查找路径")中。

例如，如果应用程序服务器 hosta：被增加应用程序查找路径，那么目录 hosta:/etc/dt/appconfig/icons/*language* 将自动地增加到图标查找路径。

图标查找路径环境变量

数据库查找路径由内置位置和下面的输入变量组合而成：

DTSPSYSICON—系统范围图标查找路径输入变量

DTSPUSERICON—个人图标查找路径输入变量

使用这些输入变量来指定在应用程序查找路径之外的位置。

已组合的数据库查找路径由两个输出变量来指定：

XMICONSEARCHPATH—用于彩色显示

XMICONBMSEARCHPATH—用于单色显示

图标查找路径输入变量的语法

变量 DTSPSYSICON 和 DTSPUSERICON 的语法是：

VARIABLE=location [,*location*...]

其中，*location* 可以有语法：

/*path*—指定本地 (会话服务器) 系统上的目录。使用此语法来添加本地目录。

要指定在另一系统上的位置，请使用它的网络文件名 —例如，/nfs/servera/projects/icons。

如何组合图标查找路径

图标查找路径的值 (XMICONSEARCHPATH 和 XMICONBMSEARCHPATH) 是通过组合下面的按优先级顺序列出的位置而创建的：

- 使用 DTSPUSERICON 变量指定的位置
- 由 DTSPUSERAPPHOSTS 变量导出的位置
- 缺省个人位置： *HomeDirectory* /.dt/icons
- 缺省位置：/etc/dt/appconfig/icons/*language*
- 使用 DTSPSYSICON 变量指定的位置
- 由 DTSPSYSAPPHOSTS 变量导出的位置
- /usr/dt/appconfig/icons/*language*

  彩色和单色的查找路径只在赋予像素图和位图的优先级方面不同。XMICONSEARCHPATH 变量将像素图列在位图之前； XMICONBMSEARCPATH 将位图列在像素图之前。

帮助查找路径

帮助查找路径指导桌面来搜索指定的文件位置，这些文件包含将要在系统中注册的帮助信息。

缺省帮助查找路径

缺省的帮助查找路径包含个人的、系统范围的和内置的位置。 缺省的 *language* 是 C。

个人位置— *HomeDirectory* /.dt/help

系统范围位置—/etc/dt/appconfig/help/*language*

内置位置—/usr/dt/appconfig/help/*language*

应用程序查找路径如何影响帮助查找路径

当向应用程序查找路径添加位置时，相应的帮助子目录将自动地添加到帮助查找路径(请参见第 139 页"应用程序查找路径如何影响数据库、图标和帮助的查找路径")。

例如，如果应用程序服务器 hosta: 被添加到应用程序查找路径， 那么目录 hosta:/etc/dt/appconfig/help/*language* 将自动添加到帮助查找路径中。

帮助查找路径环境变量

帮助查找路径是由内置的位置和下面的输入变量组合而成的：

DTSPSYSHELP—系统范围的帮助查找路径输入变量

DTSPUSERHELP—个人帮助查找路径输入变量

使用这些输入变量来指定在应用程序查找路径之外的位置。

已组合的数据库查找路径是由输出变量 DTHELPSEARCHPATH 指定的。

帮助查找路径输入变量的语法

变量 DTSPSYSHELP 和 DTSPUSERHELP 的语法是:

VARIABLE=*location* [,*location* . . .]

其中，*location* 可以有语法：

/path—指定本地 (会话服务器) 系统上的目录。使用此语法来添加本地目录。

要指定在另一系统上的位置，请使用它的网络文件名—例如，`/nfs/servera/projects/help`。

如何组合帮助查找路径

帮助查找路径的值 (`DTHELPSEARCHPATH`) 是通过组合下面的按优先级顺序列出的位置而创建的：

- 使用 `DTSPUSERHELP` 变量指定的位置
- 由 `DTSPUSERAPPHOSTS` 变量导出的位置
- 缺省个人位置： *HomeDirectory* `/.dt/help`
- 缺省目录：`/etc/dt/appconfig/help/`*language*
- 使用 `DTSPSYSHELP` 变量指定的位置
- 由 `DTSPSYSAPPHOSTS` 变量导出的位置
- `/usr/dt/appconfig/help/`*language*

本地化的查找路径

输出变量包含本地化位置和缺省 (C) 位置的项。

例如，缺省的应用程序查找路径是：

```
HomeDirectory/.dt/appmanager
/etc/dt/appconfig/appmanager/language
  /etc/dt/appconfig/appmanager/C
/usr/dt/appconfig/appmanager/language
  /usr/dt/appconfig/appmanager/C
```

其中，*language* 是 LANG 环境变量的值。

对于每个范围 (系统范围的和内置的)，指定语言的位置具有高于缺省位置的优先级。

第 10 章

操作与数据类型的介绍

操作和**数据类型**是将应用程序集成进桌面的功能强大的组成部分。它们提供了为启动应用程序和操作其数据文件而创建用户接口的一种方法。

- 第 147 页"操作的介绍"
- 第 152 页"数据类型的介绍"

本章介绍了操作和数据类型的概念。它描述了：

- 为什么要为应用程序创建操作和数据类型。
- 操作和数据类型之间如何相互关联。
- 操作和数据类型与桌面打印之间如何相互关联。

创建操作和数据类型的过程和规则在本手册的三个章节中进行介绍。

- 第 11 章 解释了如何使用桌面应用程序"创建操作"来创建各种操作和数据类型。

  可以使用"创建操作"来为大多数应用程序创建操作和数据类型而不必学习它们定义的语法规则。

- 第 12 章并第 13 章解释了如何通过创建和编辑配置文件来手动创建操作和数据类型。

  若要使用"创建操作"所不支持的高级特性，则必须手动创建操作和数据类型。

操作的介绍

操作是所写的指令，用于使桌面任务自动化，如运行应用程序和打开数据文件。操作与应用程序宏或编程函数十分相似。每个操作都具有用于运行该操作的名称。

一旦定义了操作，就可以用它对桌面用户接口进行适配，使任务更易于完成。桌面提供了将用户接口组件，例如图标、"前面板"控制和菜单项，附加到操作的能力。

例如，“应用程序管理器”中的 Desktop_Tools 应用程序组包含了启动各种公用程序的图标。



图形 10–1 Desktop_Tools 应用程序组中的操作图标

双击其中的每个图标都运行一个操作。例如，当用户双击标记为“Xwd 显示”的图标时，该操作的一部分定义将运行。该操作在配置文件 `/usr/dt/appconfig/types/`*language*`/xclients.dt` 中定义：

```
ACTION Xwud
  {
       LABEL                Xwd Display
       TYPE                 COMMAND
       EXEC_STRING         /usr/bin/X11/xwud -noclick -in \
                                %(File)Arg_1"Xwd File To Display:"%
       ...
  }
```

当用户双击图标时，将运行该操作的 `EXEC_STRING` 中的命令。

“前面板”也使用操作。例如，下面是“个人应用程序”子面板中标记为“终端”的控制的一部分定义。该控制在配置文件 `/usr/dt/appconfig/types/`*language* `/dtwm.fp` 中定义：

```
CONTROL Term
  {
       ICON                 Fpterm
       LABEL                 Terminal
       PUSH_ACTION         Dtterm
       ...
   }
```

`PUSH_ACTION` 字段指定用户单击控制时所要运行的操作—这种情况下，是名为 Dtterm 的操作。

操作的另一常见用法是在菜单中。数据文件通常在“文件管理器”的“选中”菜单中具有操作。例如，XWD 文件 (以 .xwd 或 .wd 文件名结尾的文件) 具有“打开”操作， 该操作在运行 Xwud 操作时显示屏幕图像。



图形 10–2 数据类型 XWD 文件的打开操作

在“选中”菜单中的操作是在 XWD 文件的数据类型定义里指定的。该定义位于配置文件 /usr/dt/appconfig/types/*language*/xclients.dt 中。

```
DATA_ATTRIBUTES XWD
  {
   ACTIONS         Open，Print
   ICON             Dtxwd
      ...
  }
```

XWD 数据类型，以及与其相关联的“打开”和“打印”操作，在 第 153 页“数据类型如何将数据文件连接到操作”中解释。

操作如何创建应用程序的图标

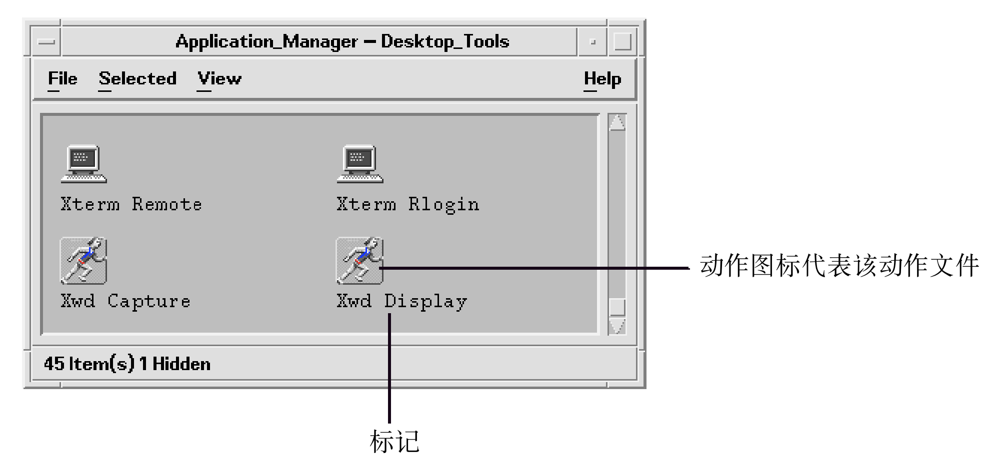
考虑在 Desktop_Tools 应用程序组中的“Xwd 显示”图标。双击此图标运行 X 客户机程序 xwud。但是，此图标并不直接表示实际的 xwud 可执行的 /usr/bin/X11/xwud。

标记为“Xwd 显示”的图标出现在应用程序组中，这是因为有一个名为 Xwud (请参见图形 10–3) 的文件在该目录中。此文件表示一个潜在其名称与文件相同—Xwud 的操作。在操作定义中，操作的名称是在关键字 ACTION 之后的名称：

```
ACTION Xwud
  {
      LABEL                Xwd Display
```

```
        TYPE                COMMAND
        WINDOW_TYPE            NO_STDIO
        EXEC_STRING            /usr/bin/X11/xwud -noclick -in \
                                %(File)Arg_1" 要显示的文件 Xwd: "%
        DESCRIPTION            The Xwd Display (Xwud) XwdDisplay action \
                            displays an xwd file that was created using the \
                            Xwd Capture (Xwd) action.It uses \
                            the xwud command.
  }
```

此文件称为**操作文件**，因为它表示一个操作。当文件是可执行文件且与操作具有相同名称时就是操作文件。其在“应用程序管理器” (或“文件管理器”) 中的图标称为**操作图标**，或**应用程序图标**，因为双击它可以启动应用程序。



图形 10–3 应用程序 (操作) 图标代表操作文件

当“应用程序管理器”检测到可执行文件时，它在操作数据库中查找看是否有任何操作，其名称与文件的名称匹配。如果找到匹配的，“应用程序管理器”就知道此文件是个操作文件。

操作文件的内容是没关系；操作文件通常包含对其桌面功能加以描述的注释。

注意：操作文件与**操作定义文件**是不相同的。**操作文件**是一个与操作具有相同名称的文件。它用于在“文件管理器” 或“应用程序管理器”中创建**应用程序图标**。 **操作定义文件**是以 *name* .dt 命名的文件，它包含了操作的定义。

一旦桌面确定文件是操作文件，潜在的操作定义就用来定义操作文件的外观和性能。

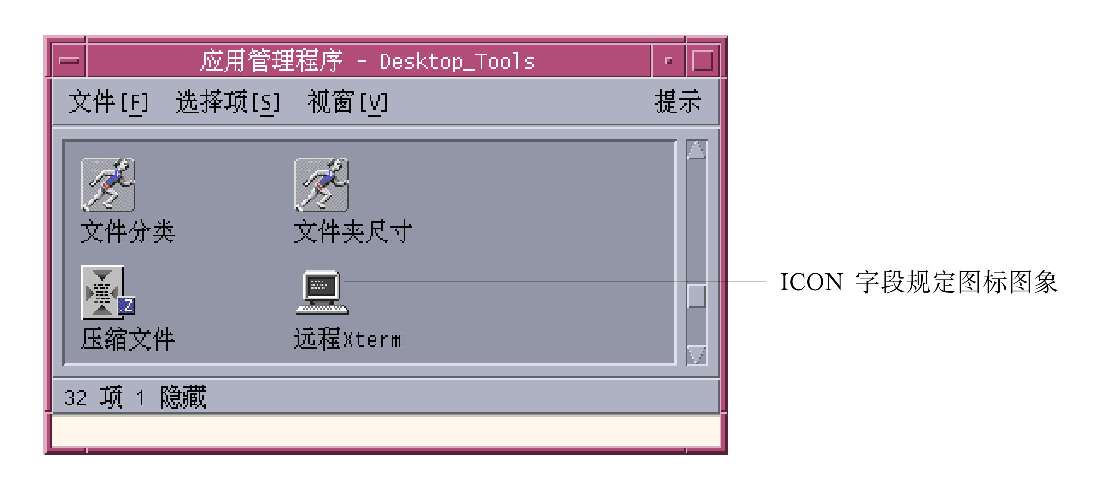
- EXEC_STRING 字段指定应用程序图标的性能。如果是“Xwd 显示”图标，EXEC _STRING 指定操作图标用特定命令行变量运行 xwud X 客户机。
- LABEL 字段指定该应用程序图标的标号。
- DESCRIPTION 字段描述用户请求“针对项”帮助时显示的文本。

- 应用程序图标使用缺省的操作图标图像，因为其操作定义中包含用来指定不同图像的 ICON 字段。

  相反，标号为“压缩文件”的图标使用不同的图标图像，因为它的潜在操作定义中包含 ICON 字段：

  例如：

```
ACTION Compress
  {
       LABEL        Compress File
       ICON        Dtcmprs
       ...
  }
```



图形 10–4 由操作定义中的 ICON 字段所指定的图标图像

Xwud 操作被称作**命令**操作，因为其定义中包含了要运行的命令 (EXEC_STRING)。操作定义中的 TYPE 字段定义了操作类型。

最初，“Xwd 显示”图标出现在 Desktop_Tools 应用程序组中。但是，在任何具有写许可的目录中可创建该操作图标的附加副本。只要 Xwud 操作定义是数据库的一部分，任何创建的名为 Xwud 的可执行文件都将成为表示该操作的操作文件，而且其在“文件管理器”或“应用程序管理器”中的图标也可以用来运行该操作。

操作如何使用作为变量的数据文件

命令的**变量** (通常是一个文件) 是命令作用到的事物。编写操作来接受文件变量。

例如，Xwud 操作的 EXEC_STRING 指定了需要一个文件变量：

```
EXEC_STRING        /usr/bin/X11/xwud -noclick -in \
          %(File)Arg_1" 要显示的 Xwd 文件："%
```

术语 Arg 代表词语**变量。** 语法 Arg_1 是指第一个变量，(File) 是指操作把变量当文件来处理。

对用户，提供文件变量的最简单的办法是将数据文件拉放到应用程序的图标上。桌面确定所拉放的文件的路径，并且将它替换到命令行中，放在 % symbols (%(File)Arg_1"要显示的 Xwd 文件："%) 之间的文本处。这样，所执行的命令为：

```
/usr/bin/X11/xwud -noclick -in file_path
```

当用户双击应用程序图标时，桌面从 EXEC_STRING 确定需要一个文件变量，并且显示对话框，提示用户输入文件名或路径。如果是 Xwud 操作，其提示为：

```
要显示的 Xwd 文件：
```

用户提供的文件名和路径用作文件变量。

操作的附加用途

除启动应用程序之外，在整个桌面范围内，操作还用于在下列项目中创建功能：

- "前面板"

  "前面板"控制的定义包括一些字段，用以指定当用户单击控制或拉放文件到其上时运行的操作。有关详细信息，请参见第 231 页"定义前面板控制"。

- 菜单

  "窗口"和"工作区"菜单定义的语法允许用菜单项指定要运行的操作。有关详细信息，请参见第 248 页"工作区管理器菜单" 和 dtwmrc(4) 手册页。

- 应用程序之间的通信

  应用程序可以设计成使用名为 ToolTalk message (TT_MSG) 的特殊类型操作来发送和接收信息。TT_MSG 操作在桌面的开发者环境文档中有描述。

数据类型的介绍

当用户创建新的数据文件时，在"文件管理器"中，文件图标的外观与性能随用户创建的数据文件类型的不同而不同。这种为文件和目录创建定制的外观和性能的能力由桌面数据分类机制提供。

什么是数据类型?

数据类型是在桌面数据库内部定义的一种结构。例如，下面是 XWD 数据类型的定义。该定义在配置文件 /usr/dt/appconfig/types/*language*/xclients.dt 中：

```
DATA_ATTRIBUTES XWD
 {
     ACTIONS            Open，Print
     ICON            Dtxwd
     NAME_TEMPLATE    %s.xwd
     MIME_TYPE          application/octet-stream
     SUNV3_TYPE        xwd-file
     DESCRIPTION      This file contains a graphics image in the XWD \
                     format. These files are typically created by \
                     taking snapshots of windows using the XwdCapture \
                    action. Its data type is named XWD. XWD files \
                    have names ending with '.xwd' or '.wd'.
 }

 DATA_CRITERIA XWD1
 {
     DATA_ATTRIBUTES_NAME            XWD
     MODE                                    f
     NAME_PATTERN                        *.xwd
 }

 DATA_CRITERIA XWD2
 {
     DATA_ATTRIBUTES_NAME            XWD
     MODE                                    f
     NAME_PATTERN                        *.wd
 }
```

每个数据类型定义均有两部分：

DATA_ATTRIBUTES—描述数据类型的外观和性能。

DATA_CRITERIA—指定将一个文件归类至所属数据类型的规则 (命名或内容)。

DATA_ATTRIBUTES_NAME 字段将标准连接到此属性。

一个 DATA_CRITERIA 可以有多个 DATA_CRITERIA。例如，XWD 数据类型具有两个标准来指定两个不同的命名标准 (NAME_PATTERN)—名称以 .xwd 或以 .wd 结尾。

数据类型如何将数据文件连接到操作

考虑 XWD 数据类型。用户通过给定文件两个文件名后缀 (扩展名)：.xwd 或 .wd 之一来创建 XWD 类型的文件。桌面使用文件名作为将文件指定为某种类型的**标准**。

XWD 数据类型通过下面的方法提供此数据类型的每个文件：

- 唯一的图标图像，用来帮助用户识别数据文件。
- "针对项"帮助，告知用户此数据类型的有关信息。
- "文件管理器"中定制的"选中"菜单包含"打开"和"打印"操作。XWD 文件的"打开"操作运行 Xwud 操作。

从“选中”菜单运行操作

“文件管理器”中的“选中”菜单，只有当文件或目录被选定时才是活动的。“选中”菜单底部的命令取决于数据类型。例如，如果选中 XWD 文件，则“选中”菜单包括“打开”和“打印”项。

数据类型定义中的 ACTIONS 字段指定了添加到数据类型的“选中”菜单底部的命令。

DATA_ATTRIBUTES XWD { ACTIONS Open，Print ... }

“选中”菜单的内容取决于数据类型。但是，许多不同的数据类型都提供“打开”操作—就是说，当在 “文件管理器”中选择特定数据类型的文件并显示“选中”菜单时，可见到“打开”命令。



图形 10–5 XWD 文件的“选中”菜单

“打开”操作通常运行与数据文件相关联的应用程序。例如，打开 XWD 文件将运行 Xwud 操作，该操作随后运行 xwud X 客户机程序来显示屏幕图像。也就是说，对于 XWD 数据类型，“打开”操作与 Xwud 操作是同义的。同样，打开数据类型 TEXTFILE 数据类型文件将运行“文本编辑器”， 而打开 BM (位图) 或 PM (像素图) 文件将运行“图标编辑器”。

创建各种完成不同事情的“打开”操作的能力使用了操作定义的两个特性：

- 操作映射

  操作映射可创建运行另一个操作的操作，而不是直接运行一命令。例如，可以创建映射到 (运行) Xwud 操作的“打开”操作。

- 对操作的数据类型限定

  操作定义可包含 ARG_TYPE 字段，该字段限定操作为某些数据类型。例如，可以指定映射到 Xwud 操作的“打开”操作只应用于 XWD 数据类型的文件。

下面是将“打开”操作映射到 XWD 数据类型 Xwud 操作的操作定义。该操作定义位于数据库配置文件 /usr/dt/appconfig/types/C/xclients.dt 中。

```
ACTION Open
  {
       LABEL                Open
       ARG_TYPE            XWD
       TYPE                MAP
       MAP_ACTION        Xwud
  }
```

TYPE 字段指定这是一个映射操作；MAP_ACTION 字段指定此操作运行 Xwud 操作。ARG_TYPE 字段指定该操作只应用于数据类型为 XWD 的文件。

比较“打开”操作的前一个定义与出现在数据库文件 /usr/dt/appconfig/types/C/dt.dt 的下一个定义中。

```
ACTION Open
  {
       LABEL                    Open
       ARG_TYPE                BM
       TYPE                    MAP
       MAP_ACTION            Dticon
  }
```

此定义应用于数据类型为 (ARG_TYPE) BM (位图文件) 的文件。该定义将“打开”操作映射到运行“图标编辑器”的 Dticon 操作。

定义数据类型的双击性能

数据类型的双击性能由 ACTIONS 字段的第一项定义。例如，对于 XWD 数据类型，双击性能将运行“打开”操作，该操作随后运行 Xwud 操作。

将数据文件拉放到操作图标上

当用户将数据文件拉放到操作图标上时，系统使用此数据文件作为操作变量来运行该操作 (请参见 第 151 页“操作如何使用作为变量的数据文件”)。

例如，当 XWD 数据文件拉放到“Xwd 显示”图标上时，使用数据文件变量运行 Xwud 操作。此操作用该数据文件来运行 xwud X 客户机程序。

为数据类型创建桌面打印

桌面打印提供打印数据文件的下列方法：

- 如果可用，在“文件管理器”的“选中”菜单中使用打印命令。
- 将数据文件拉放到桌面打印机拉放区 (“前面板”的 “打印机”控制或“打印管理器”中的打印机图标)。

除桌面打印外，许多应用程序还提供一种从应用程序内部打印的方法。

桌面打印使用名为“打印”的操作。“打印”和“打开”一样，是用于许多不同数据类型的操作名。因此，“打印”操作使用操作映射和 ARG_TYPE 字段来定制每种数据类型的打印。

例如，下面是 XWD 数据类型的“打印”操作。该定义位于 /usr/dt/appconfig/types/*language*/xclients.dt 中：

```
ACTION Print
  {
       LABEL                    Print
       ARG_TYPE                XWD
       TYPE                    MAP
       MAP_ACTION            NoPrint
  }
```

此“打印”操作专用于 XWD 文件，且映射到 NoPrint 操作。NoPrint 是在 /usr/dt/appconfig/types/*language*/dt.dt 中定义的专用操作。NoPrint 操作显示对话框，告知用户该数据类型无法打印。

将 XWD “打印”操作与下面的 PCL 文件的“打印”操作比较：

```
ACTION Print
  {
       LABEL                Print
       ARG_TYPE            PCL
       TYPE                MAP
       MAP_ACTION        PrintRaw
  }
```

在 /usr/dt/appconfig/types/*language* /print.dt 配置文件中定义的 PrintRaw 操作包含打印 PCL 文件的命令行。

```
ACTION PrintRaw
  {
   TYPE                COMMAND
   WINDOW_TYPE        NO_STDIO
   EXEC_STRING        /usr/dt/bin/dtlp -w% (File)Arg_1%
  }
```

第 11 章

用创建操作命令来创建操作和数据类型

“创建操作”是一种用于创建的工具：

- 启动应用程序的操作
- 应用程序数据文件的一种或几种数据类型
- 打开和打印应用程序数据文件的操作

对于创建简单的操作，以便运行操作系统命令和 shell 脚本，“创建操作”也是很有用的。

有关参考信息，请参见 dtcreate(1X) 的手册页。

创建操作的用途

“创建操作”包括一个主窗口和一组对话框，用来创建操作和与之相关联的数据类型。

“创建操作”有以下用途：

- 创建运行命令的操作定义。
- 创建文件 *HomeDirectory*`/.dt/types/`*action_name*`.dt`。该文件存储为应用程序创建的操作和数据类型定义。
- 在用户的起始目录下创建**操作文件**。该操作文件是与该操作同名的可执行文件。

  在“文件管理器”中，操作文件的表示方法称为**应用程序图标**，因为双击它将启动该应用程序。

  另外，在创建操作时，通过指定可拉放的数据类型，使操作图标成为拉放区。
- 为应用程序数据文件创建一种或多种数据类型 (可选)。

- 为每种数据类型创建"打开"操作。
- 为每种数据类型创建"打印"操作 (可选)。
- 重新装入操作的数据库和数据类型。这使操作和 数据类型立刻生效。

创建操作的限制

"创建操作"是为运行的应用程序创建操作和数据类型而设计的。但是，操作和数据类型是非常灵活的，并且包括只有通过手动创建定义才能访问的附加功能。

有关详细信息，请参见：

- 第 12 章
- 第 13 章

操作限制

当下面的任一条件为真时，就不能用"创建操作"为应用程序创建操作：

- 命令行需要非文件变量 (参数)。

  例如，不能用"创建操作"为下面的命令写操作：

  `lp -d`*device filename*

这里，每次执行命令用户必须提供 *device*。

- 应用程序图标必须有与操作名称不同的标记。

  例如，不能用"创建操作"来提供现有操作的本地语言版本。

- 该操作需要操作数据库的任一种高级特性。

  这些高级特性的实例是如下一些操作：

  - 从远离操作定义的系统上启动命令
  - 调用其他操作
  - 必须作为不同用户而运行 (例如，作为超级用户)
  - 广泛地使用"映射"'特性
  - 根据提供给操作的文件变量数目的不同，其行为也很不相同

数据类型限制

当下面的任一条件为真时，就不能用"创建操作"为应用程序创建数据类型：

- 该数据类型必须与除"打开"和"打印"之外的另一些操作相关联。
- 该文件类型的"打开"操作不是该操作的命令。

例如，无法用“创建操作”来创建此类数据类型，来为代表该应用程序的应用程序组的目录提供了唯一图标。

用创建操作命令为应用程序创建操作和数据类型

在运行“创建操作”之前，需要了解有关应用程序的一些知识。

- 启动应用程序的命令行。

  需要知道该命令行是否包括必需的文件变量、可选的文件变量，或者没有文件变量。

  如果该应用程序需要非文件变量，则不能用“创建操作”来创建该操作。

- 应用程序可接受的数据文件类型。

  有些应用程序只接受一种数据类型。其他应用程序 (例如 ASCII 编辑器或图形编辑器) 可以接受多种数据类型。

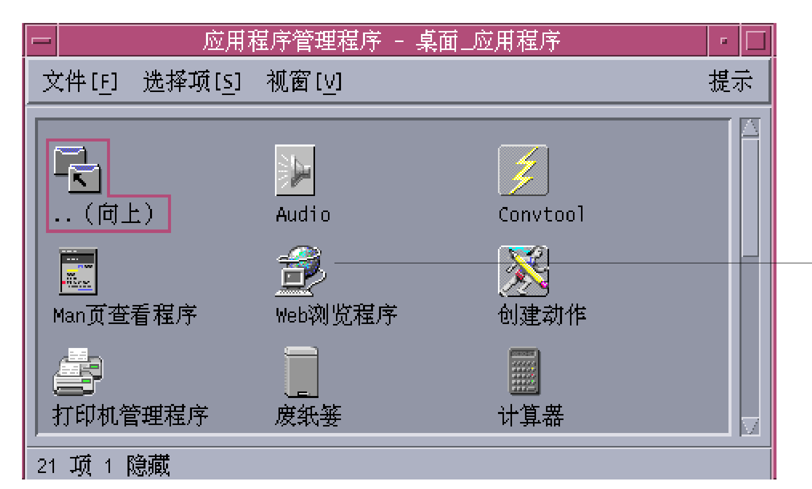
- 应用程序识别其数据文件的方式。

  可以是命名的惯例 (例如，文件名以 `.doc` 结尾)，和/或依赖于文件的内容。如果应用程序不使用命名的惯例，仍可为操作图标设置命名惯例。

- 可选：打印文件的命令行。

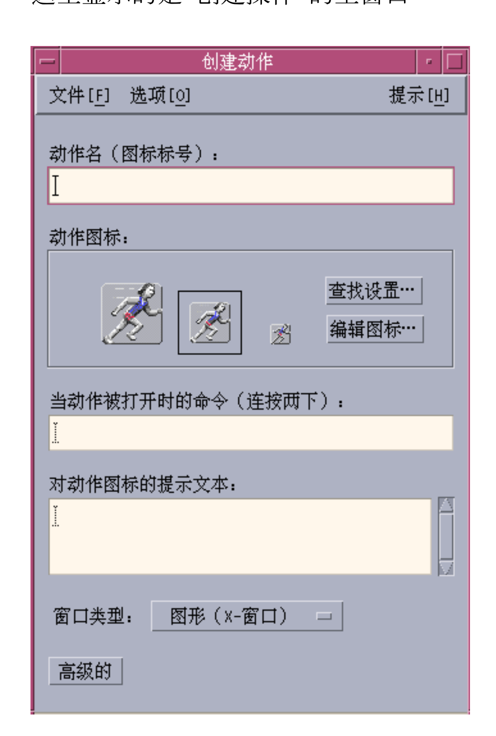
▼ 为应用程序创建操作

1. 在 Desktop_Apps 应用程序组中双击“创建操作”。



图形 11–1 “应用程序管理器”中的“创建操作”图标

这里显示的是“创建操作”的主窗口。



图形 11–2 “创建操作”主窗口

2. 在“操作名称”文本字段中，请键入将要标记操作图标的名称。

3. **使用“操作图标”控制来指定应用程序的图标。开始时，显示缺省图标。**
   - 要选择另外一个，现有的图标，单击“查找集”来显示“查找集”对话框。请参见第 167 页“使用“查找集”对话框来指定图标”。
   - 要创建新图标，请选择“编辑图标”以运行“图标编辑器”。

4. **在“操作图标打开时的命令”文本字段中，请键入启动该应用程序的命令。**

   将语法$*n* 用于文件变量；例如：

   ```
   emacs
     bitmap $1
     diff $1 $2
     lp -oraw $1
   ```

   如果命令行包含文件变量 ($*n*)，那么其操作图标将成为文件的拉放区。

   除非明确地指定使用 shell， 否则命令行无法传到 shell 。例如，下面的命令行使用 shell 处理：

   ```
   /bin/sh -c 'ps | lp'
    /bin/sh -c 'spell $1 | more'
   ```

5. **将操作图标的“针对项”帮助文本键入“操作图标的帮助文本”文本字段。**

   文本将在文本字段中自动换行。但是这些行的中断得不到联机保存。如果想指定硬换行，请使用 `\n`。

6. **从“窗口类型”选项菜单中选择操作需要的窗口支持。**

   图形 (X-Window) —应用程序创建其自身的窗口

   终端 (自动关闭) —应用程序将在终端仿真窗口中运行，在用户退出该应用程序时，该窗口自动关闭

   终端 (手动关闭) —应用程序将在终端仿真窗口中运行，并一直打开，直到用户明确地关闭它

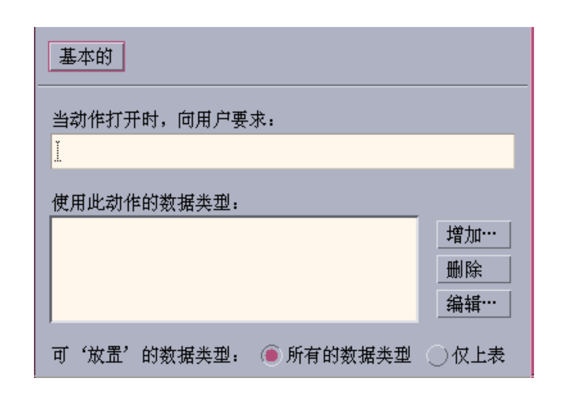
   无输出—应用程序不在显示器上产生输出

7. **按如下进行：**
   - 如果应用程序已有数据文件，并且想为其创建一种或多种数据类型，请参见下一节第 161 页“为应用程序创建一种或多种数据类型”。
   - 如果不需要创建数据类型，在“文件”菜单中，选择“保存”来保存操作。然后，在起始目录下，双击新操作的图标来对其进行测试。

▼ 为应用程序创建一种或多种数据类型

1. **用上一节第 159 页“为应用程序创建操作”所述的过程为应用程序定义操作。**

2. **在“创建操作”窗口中单击“高级”按钮来扩展窗口。**



图形 11–3 在"创建操作"主窗口中的高级特征

3. **如果想在双击应用程序图标时让它提示输入文件变量，请在"操作打开时，要求用户提供" 文本字段键入提示的文本。**

   在该文本字段中，请使用下面的准则：

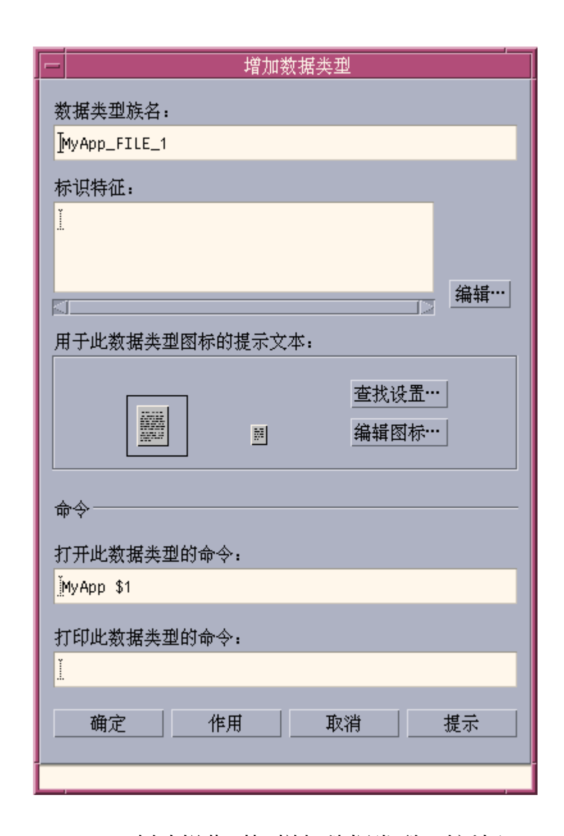
   - 如果应用程序的命令行中有**必需的**文件变量，则必须使用这个字段。
   - 如果命令行不包含文件变量，则必须让该字段保持空白。
   - 如果应用程序的命令行中的文件变量是可选的，则可以自由选择。如果提供提示文本，则在双击操作图标时将提示输入文件。如果不提供提示文本，则操作将用空字符串作为文件变量执行。

4. **指定操作将作为变量接受的文件的类型：**

   - 如果操作可以接受任一数据类型，请选择"所有数据类型"。
   - 如果操作只能接受为该应用程序创建的数据类型，请选择"仅以上列表"。

     开始时，"使用该操作的数据类型"列表是空的。当为该应用程序创建数据类型后，这些数据类型就添加到该列表中。

5. **单击"使用该操作的数据类型"列表框旁边的"增加"，来显示"增加数据类型"对话框。**

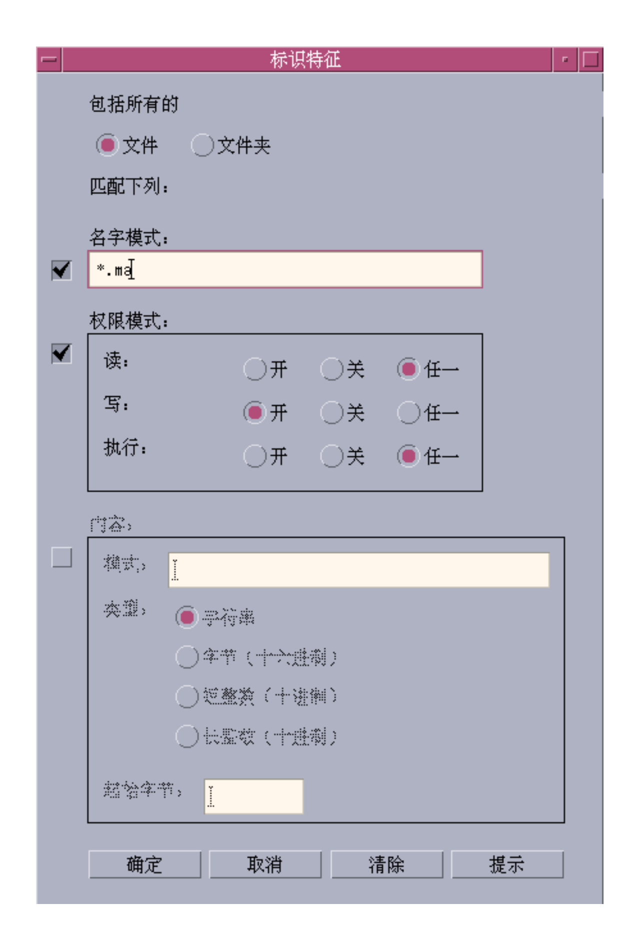


图形 11–4 “创建操作”的“增加数据类型”对话框

6. **可选：如果不想使用缺省数据类型名称，请在“数据类型族的名称”文本字段中，为该数据类型键入新名称。**

   名称不能包含空格。数据类型名称对于应用程序用户是不可见的；它用在操作／数据类型数据库中，来标识数据类型定义。

7. **单击“标识特性”框旁边的“编辑”按扭来显示“标识特性”对话框。**



图形 11–5 “创建操作”的 “识别特性”对话框

数据类型的特性是用来将该类型数据与其他类型区分开来的标准。可选择下面的一种或几种标准：

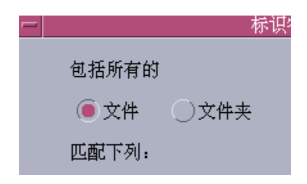
“文件”或“文件夹”—该数据类型只应用于文件或只应用于文件夹

名称模式—基于文件名的数据分类

许可模式—读、写、执行许可

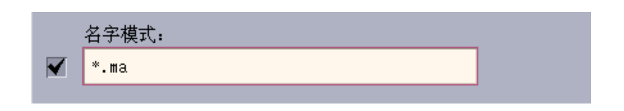
内容—文件指定部分的内容

8. 选择数据类型是代表文件还是代表文件夹。



图形 11–6 为数据类型指定文件或目录特性。

9. **如果数据分类取决于名称，选择“名称模式”复选框并在文本字段内填写。**



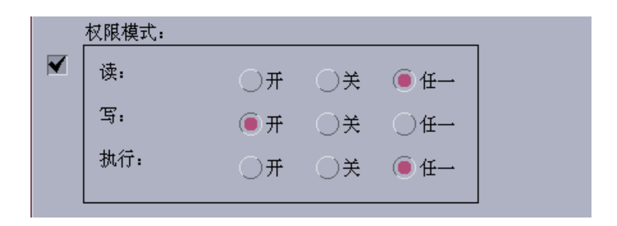
图形 11–7 为数据类型指定文件名特性

可以用 * 和 ? 作为通配符：

*—匹配任何字符序列

?—匹配任何单个字符

10. **如果数据分类取决于许可，选择“许可模式”复选框，并选择该数据类型的许可。**



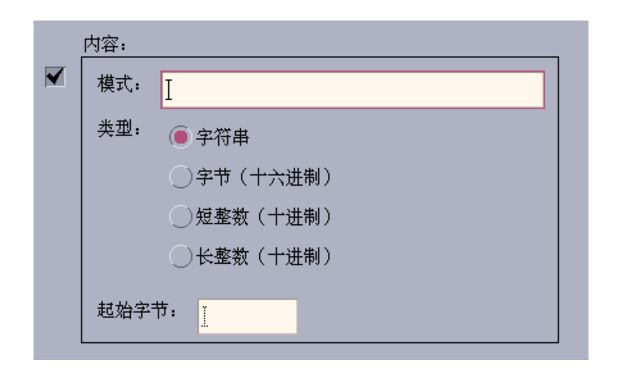
图形 11–8 为数据类型指定许可特性

开—文件必须有指定的许可

关—文件必须没有指定的许可

任意—与指定的许可没有关系

11. **如果数据分类取决于内容，则选择“内容”复选框，并提供所需的信息—要查找的模式和内容的“类型”。另外，还可以提供应开始查找的字节位置。**



图形 11–9 为数据类型指定内容特性

注意：使用基于内容的数据分类可能会影响系统的性能。

---

12. **单击“确定”来关闭“标识特性”对话框。**

    特性将在“标识特性”字段中，用下面的编码显示出来：

    d—目录

    r—文件具有读许可

    w—文件具有写许可

    x—文件具有执行许可

    !—逻辑操作符 NOT

    &—逻辑操作符 AND

13. **在“帮助文本”文本字段中，请键入数据文件的帮助文本。**

14. **使用“数据类型图标”控制来为数据文件指定图标。开始时，所显示的是缺省图标。**

    - 要选择不同的，现有的图标，单击“查找集”来显示“查找集”对话框。请参见第 167 页“使用“查找集”对话框来指定图标”。
    - 要创建新图标，单击“编辑图标”来运行“图标编辑器”。

15. **在“打开该数据类型的命令”文本字段中校验该命令。这就是双击数据文件时，所要执行的命令。**

16. **可选：从命令行，应用程序为打印数据文件提供打印命令，请使用文件变量的语法*$n* 将该命令键入 “打印该数据类型的命令” 文本字段。**

17. **执行下列操作之一来保存数据类型定义：**

    - 单击“确定”来保存数据类型，并关闭“添加数据类型”对话框。

- 单击“应用”来保存数据类型，但不关闭“添加数据类型”对话框。此后可以立即为操作定义另一种数据类型。

使用“查找集”对话框来指定图标

在“创建操作”主窗口或在“添加数据类型”窗口中单击“查找集”时将显示“查找集”对话框。用该对话框来指定用于操作或数据类型的图标。



图形 11–10 “查找集”对话框

“查找集”对话框用于指定一组有下列位置的的图标图像文件：

- 在图标查找路径的目录里。“图标文件夹”列表包含图标查找路径中的所有目录。
- 在将用 dtappintegrate 与桌面集成的登记软件包中。这些图标尚未放置于图标查找路径的目录中，但将通过 dtappintegrate 放置到那里。

注意 ：用“创建操作”创建的操作和数据类型定义为图标文件写出基本名称 (文件名去掉表示文件大小和类型的后缀)。用“创建操作”创建的操作和数据类型的图标，最后必须放置在图标查找路径的目录中。

▼ 指定一组位于图标查找路径中的图标

1. **在“查找集”对话框的“图标文件夹”列表中，双击含有该图标的文件夹路径。**

   “图标文件”列表将显示该文件夹中的所有图标文件。

2. **在“图标文件”列表中，单击要使用的图标。**

   这把图标文件的基本名称放置在“输入图标文件名”文本字段中。

3. **单击“确定”。**

▼ 在登记软件包中指定图标

如果你是创建登记软件包的系统管理员或程序员，图标图像文件最初位于登记软件包的目录中：

app_root/dt/appconfig/icons/*language*

用 dtappintegrate 登记后，图标文件将被复制到在图表查找路径中的 /etc/dt/appconfig/icons/*language*。

用此过程来指定属于登记软件包一部分的图标：

1. **在“查找集”对话框的“输入图标文件名”文本字段中，请键入图标文件的基本名称。**
2. **单击“确定”。**

   “创建操作”显示对话框，通知在图标查找路径的目录中未找到这些图标。
3. **在出现的信息对话框中，选择“不更改”。**

第 12 章

手动创建操作

创建操作有两种方法：

- 使用“创建操作”桌面应用程序
- 手动创建操作定义

手动创建操作需要编辑数据库文件。本章描述如何进行手动创建操作定义。

- 有关操作的介绍，请参见第 10 章。
- 有关使用“创建操作”的信息，请参见第 11 章。
- 有关操作定义的参考信息，请参见 dtactionfile(4) 手册页。

必须手动创建操作的理由

共有三种基本操作类型：

- `COMMAND`
- `MAP`
- `TT_MSG`

“创建操作”工具设计用来创建一些特定类型的 COMMAND 和 MAP 操作。所有 TT_MSG 操作都必须手动创建。

有关详细信息，请参见第 158 页“创建操作的限制”。

COMMAND 操作

command **操作**执行命令启动应用程序或公用程序、运行 shell 脚本命令或执行操作系统命令。 操作的定义包括要执行的命令 (EXEC_STRING)。

“创建操作”工具能用于创建最通用类型的命令操作。不过在有些情况下必须手动创建操作。例如，如果操作中有以下指定就必须手动创建 COMMAND 操作：

- 多文件变量，而且每个变量有不同的提示符。
- 操作调用—操作调用其他操作的能力。
- 变量计数依赖性能—创建对不同文件变量数目而有很不相同性能的操作的能力。
- 远程执行主机—在包含操作定义的系统之外的系统上运行应用程序的能力。
- 用户的更改—作为不同用户运行该操作的能力 (例如，提示输入超级口令， 然后以超级用户身份运行操作)。

MAP 操作

映射操作是不直接指定命令或 ToolTalk 消息，而是“映射 ”到另一个操作的操作

映射可以指定操作的替代名称。例如，名为 IconEditor 的内置命令操作启动“图标编辑器”。数据库还包括对位图和像素图文件的定义中 (通过 ARG_TYPE 字段) 限定的 Open 操作，它映射到 IconEditor 操作。这使用户可以通过在“文件管理器”中选择位图或像素图文件并在“选中”菜单中选择“打开”来启动“图标编辑器”。

“创建操作”能为“打开”和“打印”操作提供有限的映射功能。所有其他的映射操作必须手动创建。

TT_MSG (ToolTalk 消息) 操作

TT_MSG 操作发送 ToolTalk 消息。所有 TT_MSG 操作都必须手动创建。

手动创建操作：一般步骤

本节介绍如何为操作定义创建配置文件。

操作的配置文件

包含操作定义的配置文件必须满足下面的要求：

- 文件必须使用命名惯例*name*`.dt`
- 文件必须位于数据库 (操作和数据类型) 的查找路径内。缺省的查找路径为：
  个人操作—*HomeDirectory* `/.dt/types`
  系统范围操作—`/etc/dt/appconfig/types/`*language*
  内置操作—`/usr/dt/appconfig/types/`*language*。你不应使用此目录。

有关修改操作/数据类型查找路径的信息，请参见第 136 页"设置查找路径的值"。

▼ 手动创建操作

1. **打开现有的数据库文件或创建新文件。**
   请参见上一节第 171 页"操作的配置文件"。
2. **使用如下语法创建操作定义：**

```
ACTIONaction_name
    {
     TYPE    action_type
     action_field
             ...
        }
```

其中：

action_name—运行操作所用的名称。

action_type—COMMAND (缺省值) 、MAP 或 TT_MSG

action_field—这类操作必需或任选的字段之一。所有字段都由关键字和值组成。

本章包含多种操作字段。有关详细信息，请参见 `dtactionfile` (4) 手册页。

3. **保存文件。**
4. **如果需要操作图标有唯一的图像，请为操作创建图标。图标的缺省位置为：**
   - 个人图标：*HomeDirectory* `/.dt/icons`
   - 系统范围图标：`/etc/dt/appconfig/icons/`*language* 。缺省的 *language* 是 C。

有关详细信息，请参见第 175 页"指定"操作所用的图标图像""。

5. **在 Desktop_Tools 应用程序组中双击"重新装入操作"。**

6. **为操作创建操作文件。该操作文件在"文件管理器"或"应用程序管理器 "中创建代表该操作的图标。(如果操作被写以启动应用程序，则该图标称为应用程序图标。)**

   要创建操作文件，需要创建与 *action_name* 同名的可执行文件。该文件可位于任何您具有写许可的目录内。可创建任意多个操作文件。

创建 COMMAND 操作的实例

下面的步骤在远程系统 AppServerA上创建了启动传真应用程序的个人操作。启动传真应用程序的命令是：

`/usr/fax/bin/faxcompose` [*filename*]

1. **创建** *HomeDirectory*`/.dt/types/Fax.dt` **文件。**

2. **将下面的操作定义放到文件中：**

```
ACTION FaxComposer
    {
      TYPE              COMMAND
      ICON              fax
      WINDOW_TYPE    NO_STDIO
      EXEC_STRING    /usr/fax/bin/faxcompose -c%Arg_1%
      EXEC_HOST      AppServerA
      DESCRIPTION    Runs the fax composer

    }
```

   `WINDOW_TYPE` 和 `EXEC_STRING` 字段描述操作的性能。

   `WINDOW_TYPE`—`NO_STDIO` 关键字指定该操作不必在终端仿真器窗口中运行。

   请参见第 184 页"为操作指定窗口支持"。

   `EXEC_STRING`—语法 `%Arg_1%` 接受拉放文件。如果双击操作图标，操作将打开空的传真书写器窗口。

   请参见第 178 页"生成 COMMAND 操作的执行字符串"。

3. **保存文件。**

4. **用"图标编辑器"在** *HomeDirectory* `/.dt/icons` **目录中创建下面的图标图像文件：**

   - fax.m.pm，大小为 32 乘 32 像素
   - fax.t.pm，大小为 16 乘 16 像素

5. **在 Desktop_Tools 应用程序组中双击"重新装入操作"。**

6. **在具有写许可的目录 (例如，起始目录) 中创建名为** `FaxComposer` **的可执行文件。**

创建 MAP 操作的实例

假设绝大部分传真文件用“文本编辑器”创建，而且属于数据类型 TEXTFILE (文件名为 *.txt)。

下面的步骤在数据类型的“选中”菜单中加入“传真”菜单项。

1. **打开在上例中创建的** *HomeDirectory*/.dt/types/Fax.dt **文件。**
2. **在文件中加入下面的映射操作的定义：**

```
ACTION Fax
  {
   ARG_TYPE       TEXTFILE
   TYPE       MAP
   MAP_ACTION      FaxComposer
  }
```

3. **保存文件。**
4. **将 TEXTFILE 的数据属性定义从** /usr/dt/appconfig/types/*language* /dtpad.dt **复制到新的文件** *HomeDirectory* /.dt/types/textfile.dt**。将“传真”操作添加到** ACTIONS **字段。**

```
DATA_ATTRIBUTES TEXTFILE
  {
   ACTIONS     Open，Print，Fax
   ICON       Dtpenpd
   ...
  }
```

5. **保存文件。**
6. **打开应用程序管理器，在 Desktop_Tools 应用程序组中双击“重新装入操作”。**

▼ 重新装入操作／数据类型数据库

为使新的或经过编辑的操作定义有效，桌面必须重新读取数据库。

- **打开 Desktop_Tools 应用程序组，双击“重新装入操作”。**
- **或者，执行下面的命令：**

```
dtaction ReloadActions
```

ReloadActions 是图标的标记为“重新装入操作”的操作的名称。

当用户进行下面的操作时，操作数据库也被重新读取：

- 登录
- 重新启动“工作区管理器”
- 从“文件”菜单选择“保存”在“创建操作”窗口保存操作

为“操作”创建“操作文件”(图标)

操作文件是为在“文件管理器”或“应用程序管理器” 中提供操作的可视表示而创建文件。



图形 12–1 “应用程序管理器”中的操作文件 (操作图标)

由于操作文件的图标表示一个操作，因此有时也被称为**操作图标**。如果其潜在的操作启动应用程序，则操作文件图标也被称为**应用程序图标**。

双击操作图标来运行操作。操作图标同时也可能是拉放区。

▼ 创建操作文件 (操作图标)

● **创建与操作同名的可执行文件。该文件的内容并不重要。**

例如，操作的定义如下：

```
ACTION  MyFavoriteApp
 {
   EXEC_STRING      Mfa -file%Arg _1%
   DESCRIPTION      Runs MyFavoriteApp
   ICON                Mfapp
 }
```

此时，操作文件是名为 `MyFavoriteApp` 的可执行文件。在“文件管理器”和“应用程序管理器”中，`MyFavoriteApp` 文件将使用图标图像 `Mfapp.`*size*`.`*type*。双击 `MyFavoriteApp` 的图标就运行该操作的执行字符串，图标的“针对项”帮助则为 `DESCRIPTION` 字段 (“运行 `MyFavoriteApp`”) 的内容。

“操作标号”

如果操作定义包含 LABEL 字段，该操作文件就在“文件管理器”和“应用程序管理器”中加上该字段内容的标号而不是文件名 (*action_name*)。例如，如果操作定义中包含下面的定义：

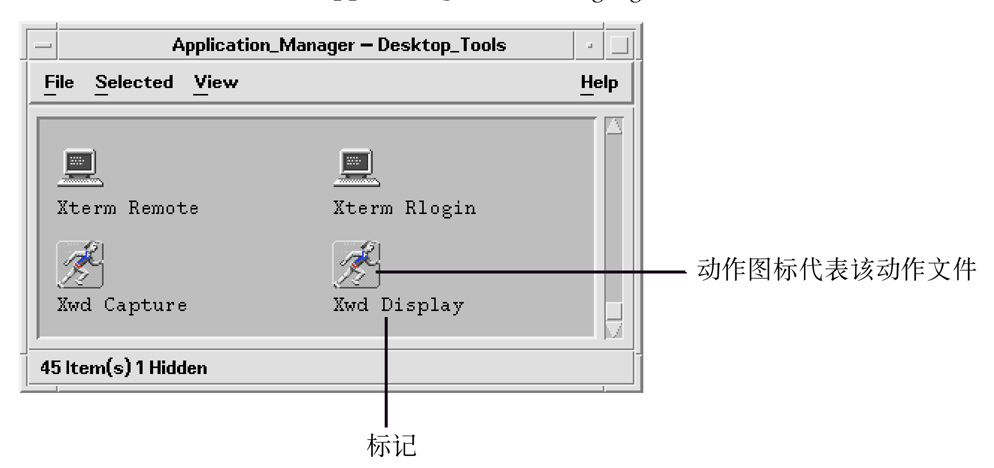
```
ACTION  MyFavoriteApp
 {
   LABEL      Favorite Application
   ...
 }
```

此时操作图标标记为“喜爱的应用程序”。

指定“操作所用的图标图像”

使用 ICON 字段来指定用于“文件管理器”和“应用程序管理器”的图标，作为为操作创建的操作图标。

如果未指定图标，系统将使用缺省的操作图标图像文件 /usr/dt/appconfig/icons/*language* /Dtactn.*。



图形 12–2 缺省的操作图标图像

缺省的操作图标可使用下面的资源加以更改：

*actionIcon: *icon_file_name*

其中 *icon_file_name* 可以是基本名称或绝对路径。

ICON 字段的值可以是：

- 基本文件名

基本文件名是包含图标图像的全文件名减去有关大小 (m 和 t) 和图像类型 (bm 和 pm) 的后缀。例如，如果全文件名为 GameIcon.m.pm 和 GameIcon.t.pm，则基本文件名为 GameIcon。

如果采用基本文件名，图标文件必须放置在位于图标查找路径的目录内：

- 个人图标：*HomeDirectory* /.dt/icons
- 系统范围图标：/etc/dt/appconfig/icons/*language*

- 图标文件的绝对路径，包含全文件名

  只有在图标文件不在图标查找路径内时才使用绝对路径。例如，如果图标文件 GameIcon.m.pm 放在目录 /doc/projects (不在图标查找路径) 下，ICON 字段值则为 /doc/projects/GameIcon.m.pm。

  表 12–1 列出了应该创建的图标大小和相应的文件名。

表 12–1 操作图标的图标名和图标大小

像素大小	位图名	像素图名
48 乘 48	*name* .l.bm	*name* .l.pm
32 乘 32	*name* .m.bm	*name* .m.pm
16 乘 16	*name* .t.bm	*name* .t.pm

▼ 修改现有的操作定义

可修改系统内任何可用的操作，包括内置操作。

注意：修改内置操作数据库时必须小心。因为内置操作是为很好使用桌面应用程序而设计的。

---

1. **找到需要修改的操作定义。**

   操作定义的缺省位置是：

   - 内置操作：/usr/dt/appconfig/types/*language*
   - 系统范围操作：/etc/dt/appconfig/types/*language*
   - 个人操作：*HomeDirectory* /.dt/types

     系统也可能包括附加的位置。要查看系统中所用操作的位置列表，请键入下面的命令：

     dtsearchpath -v 系统所用目录列在 DTDATABASESEARCHPATH 下。

2. **如果必要的话，将操作定义的文本复制到下面的目录之一中新的或现有的文件中：**

   - 系统范围操作：/etc/dt/appconfig/types/*language*
   - 个人操作：*HomeDirectory* /.dt/types

由于不应在 /usr/dt/appconfig/types/*language* 目录中编辑文件，所以必须复制内置操作。

3. **编辑操作定义。**
4. **编辑完毕后请保存文件。**
5. **在 Desktop_Tools 应用程序组中双击“重新装入操作”。**

操作定义中的优先级

当用户调用操作时，系统在数据库中查找匹配的操作名。如果存在多个由该名称的操作，系统则利用优先级规则来决定使用哪个操作。

- 如果没有其他优先级规则可用，将使用基于定义位置的优先级。下表按从高到低的优先级排列：
  - 个人操作 (*HomeDirectory* /.dt/types)
  - 系统范围的本地操作 (/etc/dt/appconfig/types/*language*)
  - 系统范围的远程操作 (*hostname* :/etc/dt/appconfig/types/*language*)。只查找在应用程序查找路径内的远程主机。
  - 内置操作 (/usr/dt/appconfig/types/*language*)
- 在给定的目录内，*.dt文件按字母顺序读取。
- 由 ARG_CLASS、ARG_TYPE、ARG_MODE 或 ARG_COUNT 限定的操作优先于未限定的操作。(上述四种字段的缺省值是*。)

  当应用了多个限定时，优先级顺序从高到低是：

  - ARG_CLASS
  - ARG_TYPE
  - ARG_MODE
  - ARG_COUNT

    存在多于一个限定的 ARG_COUNT 时，优先级顺序从高到低是：

  - 特定整数值 *n*
  - <*n*
  - >*n*
  - *

  例如，考虑操作定义的下面部分：

```
ACTION EditGraphics
   # EditGraphics-1
   {
     ARG_TYPE    XWD
     ...
   }
```

```
ACTION EditGraphics
   # EditGraphics-2
   {
     ARG_COUNT     0
     ...
   }

ACTION EditGraphics
   # EditGraphics-3
   {
     ARG_TYPE     *
     ...
   }
```

因为没有提供变量并且 ARG_COUNT 0 优先，双击 EditGraphics 操作图符启动 EditGraphics-2。当提供了 XWD 类型的文件变量时，使用 EditGraphics-1，因为它指定了 XWD ARG_TYPE。对于所有其他文件的变量，则使用 EditGraphics-3。

生成 COMMAND 操作的执行字符串

COMMAND 操作至少需要两个字段—ACTION 和 EXEC_STRING。

```
ACTION action_name
 {
    EXEC_STRINGexecution_string
 }
```

执行字符串是 COMMAND 操作定义中最重要的部分。它使用的语法与在“终端”窗口内执行的命令行相似，但包含了用于处理文件和字符串变量的附加语法。

执行字符串的一般特性

执行字符串可能包括：

- 文件和非文件变量
- shell 语法
- 可执行文件的绝对路径或名称

“操作变量”

变量是命令或应用程序正确运行所需要的信息。例如，考虑可用于在“文本编辑器”中打开文件的命令：

dtpad*filename*

在该命令中，*filename* 是 dtpad 命令的文件变量。

象应用程序和命令一样，操作也可以有变量。COMMAND 操作可以使用两种类型的数据：

- 文件
- 字符串数据

在“执行字符串”中使用 shell

执行字符串不经过 shell 就直接执行。但是，也可以明确地在执行字符串中调用 shell。

例如：

```
EXEC_STRING                \
   /bin/sh -c \
   ’tar -tvf% (File)Arg_1% 2>&1 | \${PAGER:-more;\
   echo “\\n*** Select Close from the Window menu to close ***”’
```

可执行文件的名称或绝对路径

如果应用程序位于 PATH 变量所列的目录之中，则可以使用简单的可执行文件名。如果应用程序在别处，则必须使用可执行文件的绝对路径。

创建不使用变量的操作

从命令行启动应用程序，使用相同的 EXEC_STRING 语法。

实例

- 下面的执行字符串是启动 X 客户机程序 xcutsel 的操作的一部分。

  ```
  EXEC_STRING xcutsel
  ```

- 下面的执行字符串启动带数字式时钟的客户程序 xclock。该命令行包含命令行选项，但不需要变量。

  ```
  EXEC_STRING xclock -digital
  ```

创建接受拉放文件的操作

文件变量使用如下语法：

%Arg_*n*%

或

`%(File)Arg_n%`

(File) 是可选项，因为 Arg_*n* 提供的变量 (缺省) 假定为文件。(请参见第 181 页"将文件变量解释成字符串"，以获得% (String)Arg_*n*% 语法的使用方法。)

该语法使用户可以通过在操作图标上拉放数据文件对象来启动带有该文件变量的操作。它将第 *n* 个变量替代到命令行中。拉放文件可以是本地文件或远程文件。

实例

- 下面的执行字符串执行使用拉放文件作为-装入参数的 wc -w。

  ```
  EXEC_STRING wc -w%Arg _1%
  ```

- 下例所显示的是仅用目录变量工作的操作的定义的一部分。当目录被拉放在操作图标上时，该操作将该目录内所有有读写许可的文件列表显示出来。

  ```
  ACTION  List_Writable_Files
      {
        ARG_TYPE     FOLDER
        EXEC_STRING  /bin/sh -c 's -l%Arg _1% | grep rw-'
        ...
      }
  ```

创建提示输入文件变量的操作

文件变量使用如下语法：

`%(File) "提示"%`

该语法创建了一个操作，该操作在用户双击操作图标时显示提示输入文件名。

例如，下面的执行字符串显示对话框，提示输入 wc -w 命令的文件变量：

```
EXEC_STRING wc -w% (File)"Count words in file:"%
```

创建接受拉放文件或提示输入文件的操作

文件变量使用如下语法：

`%Arg_n"提示"%`

或

`%(File)Arg_n"提示"%`

该语法产生的操作：

- 接受拉放文件作为文件变量。

- 用户双击操作图标时显示对话框，提示输入文件名。

  例如，下面的执行字符串在拉放文件上执行 `lp -oraw` 操作。如果操作由双击图标启动，则出现对话框提示输入文件名。

  ```
  EXEC_STRING lp -oraw%Arg _1"File to print:"%
  ```

创建提示输入非文件变量的操作

对非文件变量使用如下语法：

`%"`**提示**`"%`

或

`%(String) "`**提示**`"%`

由于缺省情况下，引号内的文本被解释为字符串，所以 `(String)` 是可选项。该语法显示对话框，提示输入非文件数据；如果提示输入文件名就不能用该语法。

例如，下面的执行字符串运行 `xwd` 命令，并提示要增加到每个像素的值：

```
EXEC_STRING xwd -add% "增加值："% -out%Arg _1"文件名："%
```

将文件变量解释成字符串

变量使用如下语法：

`%(String)Arg_`*n*`%`

例如，下面的执行字符串用于打印具有包含文件名标题的文件，所用命令为 `lp-t` **标题文件名**。

```
EXEC_STRING lp -t%(String)Arg_1%% (File)Arg_1"File to
print:"%
```

在操作中提供 shell 功能

在执行字符串中指定 shell：

```
/bin/sh -c 'command'
/bin/ksh -c 'command'
/bin/csh -c 'command'
```

实例

- 下面的执行字符串说明使用 shell 管道传递的操作。

  ```
  EXEC_STRING /bin/sh -c 'ps | lp'
  ```

■ 下例的执行字符串更复杂，需要有 shell 处理并接受文件变量。

```
EXEC_STRING /bin/sh -c 'tbl%Arg _1"手册页："% | troff -man'
```

■ 下面的执行字符串要求变量为压缩文件。此操作解压缩文件并用 lp -oraw 打印该文件。

```
EXEC_STRING
   /bin/sh -c 'cat%Arg _1 "要打印的文件："% | \
     uncompress | lp -oraw'
```

■ 下面的执行字符串启动 shell 脚本。

```
EXEC_STRING /usr/local/bin/StartGnuClient
```

创建多文件变量的 COMMAND 操作

操作有三种处理多文件变量的方法：

■ 该操作可以重复运行，每一变量执行一次。当 EXEC_STRING 包含单个文件变量，而且通过在操作图标上拉放多个文件来提供多文件变量时，则对每个文件变量分别运行此操作。

例如，如果为下面的操作定义提供多文件变量，则：

```
ACTION DisplayScreenImage
    {
      EXEC_STRING   xwud -in%Arg _1%
      ...
    }
```

DisplayScreenImage 操作被重复运行。

■ 操作可以使用两个或两个以上的不可互换的文件变量。例如：

xsetroot -cursor*cursorfile maskfile*

需要特定顺序的两个专用文件。

■ 操作可以对每个文件变量顺序执行同一命令。 例如：

pr *file* [*file* ...]

将在打印作业中打印一个或多个文件。

为不可互换变量创建操作

采用下面的语法惯例之一：

■ 如果需要操作来提示输入文件名，则每个文件变量均使用如下语法：

```
%(File) "提示"%
```

对每个变量使用不同的**提示**字符串。

例如，下面的执行字符串提示输入两个文件。

```
EXEC_STRING
xsetroot -cursor% (File)"光标位图："% \
  %(File)"掩码位图："%
```

- 要接受拉放文件，为每个文件变量使用如下语法：

  `%Arg _n%`

  为每个变量使用不同的 *n* 值。例如：

  ```
  EXEC_STRING              diff%Arg _1%%Arg _2%
  ```

用“可交换的文件变量”来创建操作

采用下面的语法惯例之一：

- 要创建接受拉放文件并以 *command file 1 file 2* …形式发出命令的操作，文件变量使用如下语法：

  `%Args%`

- 要创建接受多重拉放文件，或在双击时提示输入单个文件的操作，文件变量使用如下语法：

  `%Arg_1"提示"%%Args%`

  操作采用 *command file 1 file 2* … 形式来发出命令。

实例

- 该执行字符串创建一个操作，该操作执行：

  pr *file 1 file 2*

  具有多文件变量。

  ```
  EXEC_STRING pr%Args%
  ```

- 本执行字符串创建的操作与上例相似，只不过操作在双击时显示提示符 (无文件变量)。

  ```
  EXEC_STRING  pr%Arg _1"要打印的文件："%%Args%
  ```

创建接受多重拉放文件的操作

要接受多重拉放文件变量并以下面的形式执行命令行：

command file 1 file 2 . . .

使用如下语法：

```
%Args%
```

实例

- 该执行字符串对多个文件执行名为 Checkout 的脚本：

```
EXEC_STRING                /usr/local/bin/Checkout \
                             %Arg_1"检查出什么文件? "% %Args%
```

- 本执行字符串对多个文件执行 lp -oraw 命令：

```
EXEC_STRING        lp -oraw%Arg _1"要打印的文件："%%Args%
```

用于 COMMAND 操作的窗口支持和终端仿真器

COMMAND 操作有几种桌面上支持窗口的方法。

- 如果应用程序有自己的窗口，操作可以写成不提供附加的窗口支持。该选项也用于运行不需要用户直接输入，也没有输出的命令。
- 如果应用程序必须在终端仿真器窗口内运行，操作可写成打开窗口，然后运行该应用程序。 有几种终端选项。

为操作指定窗口支持

使用 WINDOW_TYPE 字段指定如表 12–2所示的操作所需的窗口支持类型。

表 12–2 WINDOW_TYPE 字段和提供的窗口支持

WINDOW_TYPE	**提供的窗口支持**
NO_STDIO	无。如果应用程序有自己的窗口，或命令无可见输出，使用 NO_STDIO 选项。
PERM_TERMINAL	永久的终端仿真器窗口。此操作可打开终端窗口，该窗口在用户明确地关闭之前保持打开。用户可以在窗口中输入数据。用于获取某些输入，产生某些输出，然后终止的命令 (例如 ls **目录**)。
TERMINAL	临时的终端仿真器窗口。该操作打开终端窗口，而该窗口在命令完成后立即关闭。用于全屏幕命令 (例如，vi)。

为终端仿真器指定命令行选项

使用操作定义中的 TERM_OPTS 字段指定终端仿真器的命令行选项。

例如，下面的操作提示输入执行主机：

```
ACTION OpenTermOnSystemUserChooses
  {
    WINDOW_TYPE        PERM_TERMINAL
    EXEC_HOST          %(String)"远程终端在："%
    TERM_OPTS          -title% (String)"窗口标题："%
    EXEC_STRING        $SHELL
  }
```

指定不同的缺省终端仿真器

操作所使用的缺省终端仿真器是 dtterm。可以将其更改为另一终端仿真器。如果没有明确地指定要用的终端仿真器，将使用缺省终端仿真器。

操作所使用的终端仿真器必须有下面的命令行选项：

- -title *window_title*
- -e *command*

  操作使用的缺省终端仿真器由两类资源决定：

- localTerminal 资源指定本地应用程序所使用的终端仿真器。

  ```
  *localTerminal:        terminal
  ```

  例如：

  ```
  *localTerminal:                xterm
  ```

- remoteTerminal 资源指定远程应用程序所使用的终端仿真器。

  ```
  *remoteTerminal:        host:terminal  [,host:terminal...]
  ```

  例如：

  ```
  *remoteTerminal:        sysibm1:/usr/bin/xterm，syshp2:/usr/bin/yterm
  ```

限定操作使用某些变量

限定操作使用某种特定类型的变量可以改善操作的性能。例如，应限定调用 PostScript 文件的查看器的操作只使用 PostScript 文件变量；使用此限定后，在指定非 PostScript 文件时，操作将会返回错误对话框。

可以基于下面的方式来限定操作：

- 文件变量的数据类型。
- 文件变量的数量—例如，没有变量与一个或多个变量。这为操作图标提供了不同的拉放置和双击性能。
- 变量的读／写模式。

限定操作使用指定的数据类型

使用 ARG_TYPE 字段指定使操作有效的数据类型。使用数据的属性名。

可以输入一列数据类型；不同项之间用逗号隔开。

例如，下面的操作定义假定已创建了 GIF 数据类型。

```
ACTION Open_Gif
  {
     TYPE        COMMAND
     LABEL       "Display Gif"
     WINDOW_TYPE     NO_STDIO
     ARG_TYPE      Gif
     ICON       xgif
     DESCRIPTION      Displays gif files
     EXEC_STRING      xgif
   }
```

根据变量数量限定操作

使用 ARG_COUNT 字段指定此操作可接受的变量个数。有效值为：

* (缺省值) —任意个数的变量。其他值的优先级都比 * 高。

n—任意非负整数，包括 0。

>*n*—多于 *n* 个变量。

<*n*—少于 *n* 个变量。

ARG_COUNT 的一个用途是提供不同的操作图标性能，取决于用户是双击图标还是在图标上拉放一个文件。请参见下一节，“提供不同的双击和拉放性能”。

▼ 提供不同的双击和拉放特性

按此步骤来创建接受拉放文件的操作，但当双击操作图标时，并不提示输入文件。

1. **为双击功能创建操作定义。**

   使用 ARG_COUNT 字段指定 0 变量。为 EXEC_STRING 使用的语法不接受拉放文件。

2. **为放置功能创建第二个操作定义。**

使用 ARG_COUNT 字段指定 0 变量。 为 EXEC_STRING 使用的接受拉放文件。

例如，假设下面的两个命令行能用于启动名为 vedit 的编辑器：

- 不带文件变量启动编辑器：

  ```
  vedit
  ```

- 带文件参数启动编辑器，文件以只读方式 打开：

  vedit -R *filename*

下面的两个操作为名为 Vedit 的操作创建拉放和双击功能。当查找数据库以查找匹配项时，第一个操作优先，这是因为，ARG_COUNT 0 比隐含的 ARG_COUNT * 有更特定的拉放功能性定义。

```
# Double-click functionality
 ACTION Vedit
 {
     TYPE                          COMMAND
     ARG_COUNT                  0
     WINDOW_TYPE                  PERM_TERMINAL
     EXEC_STRING                  vedit
 }

 # Drop functionality
 ACTION Vedit
 {
     TYPE                          COMMAND
     WINDOW_TYPE                  PERM_TERMINAL
     EXEC_STRING                  vedit -R %Arg_1%
 }
```

根据变量的模式限定操作

使用 ARG_MODE 字段来指定变量的读/写模式。有效值为：

* (缺省值) —任意模式

!w—不可写

w—可写

创建在远程系统上运行应用程序的操作

在讨论操作和远程执行时经常要使用的两个术语：

数据库主机—包含操作定义的系统

执行主机—运行可执行文件的系统

在大多数情况下，操作和它们的应用程序位于同一系统上；由于操作的缺省执行主机与数据库主机相同，因此不需要特殊的语法。

但是，当执行主机与数据库主机不同时，操作定义必须指定执行字符串在哪个主机上运行。

在不同系统上定位操作和应用程序的能力是桌面的客户／服务器体系结构的一部分。有关网络应用程序更深层次的讨论，请参见第 122 页"管理应用程序服务"。

创建运行远程应用程序的操作

使用操作定义中的 EXEC_HOST 字段指定应用程序的位置。

EXEC_HOST 的有效值是：

%DatabaseHost%—定义了操作的主机。

%LocalHost%—调用了操作的主机 (**会话服务器**)。

%DisplayHost%—运行 X 服务器的主机 (对 X 终端不允许)。

%SessionHost%—运行控制"登录管理器"的主机。

hostname—命名的主机。 当操作只能在一台特定主机上调用时，为环境使用此值。

%"**提示** "%—在每次调用操作时，提示用户输入主机名。

缺省值为 %DatabaseHost%，%LocalHost%。因此，当忽略了 EXEC_HOST 字段，该操作先尝试在包含操作定义的主机上运行该命令。如果失败，操作再试图在会话服务器上运行命令。

实例

- 此字段指定主机 ddsyd：

```
EXEC_HOST   ddsyd
```

- 此字段提示输入主机名：

```
EXEC_HOST%  "包含应用程序的主机："%
```

- 此字段指定操作将试图在包含操作定义的主机运行应用程序。如果失败，操作将试图在主机 ddsyd 上运行应用程序。

```
EXEC_HOST%DatabaseHost%  ， ddsyd
```

在操作和数据类型定义中使用变量

在操作和数据类型的定义文件中，可以包括字符串变量和环境变量。

在操作中使用字符串变量

字符串变量定义只在定义位置至文件尾之间有效。数据库没有全局的字符串变量。

如果字符串变量和环境变量同名，字符串变量的优先级高。

▼ 定义字符串变量

- **使用如下语法：**

  set *variable_name=value*

变量名可以包含任意字母字符和下划线 (_)。 每个变量的定义必须单独占一行。

例如：

```
set Remote_Application_Server=sysapp set Remote_File_Server=sysdata
```

▼ 引用字符串变量

- **使用如下语法：**

  $[{] *variable_name* [}]

例如：

```
 EXEC-HOST            $Remote_Application_Server
CWD                    /net/${Remote_File_Server/doc/project
```

在操作和数据类型中使用环境变量

- **使用如下语法来引用环境变量：**

  $[{] *variable* [}] .

变量是在装入数据库时展开的 (被其值所取代)。如果字符串变量和环境变量同名，字符串变量的优先级高。

例如，此执行字符串打印具有登录名标题的文件。

```
EXEC-STRING lp -t$LOGNAME% (File)Arg_1%
```

从命令行调用操作

桌面提供 dtaction 命令来从命令行运行操作。可使用 dtaction 从下面的途径运行操作：

- 脚本
- 其他操作
- 终端仿真器命令行

dtaction 语法

```
dtaction [-user user_name]
[-execHost hostname] \
 action_name [argument [argument]...]
```

-user *user_name*—提供以不同用户身份运行操作的能力。如果 dtaction 被 *user_name* 以外的用户调用，将显示提示符指示输入口令。

-execHost *hostname*—仅适用于 COMMAND 操作；指定将要运行命令的主机。

argument—操作的变量；通常是文件变量。

dtaction 客户机程序还有其他的命令行选项。有关详细信息，请参见dtaction (1) 手册页。

创建运行另一个操作的操作

使用 EXEC_STRING 操作中的 dtaction。

例如，下面的操作使用名为 "Spell" 的内置操作 (该操作在"应用程序管理器"中标记为"检查拼写")。 新的操作运行"文本编辑器"和"拼写"操作，并在单独的终端仿真器窗口内显示拼写错误。

```
ACTION EditAndSpell
 {
     WINDOW_TYPE                  NO_STDIO
```

```
    EXEC_STRING                /bin/sh -c 'dtaction Spell \
       %Arg_1"文件："%; dtpad%Arg _1%'
}
```

创建作为不同的用户来运行的操作

在 EXEC_STRING 使用下列语法：

EXEC_STRING   dtaction -user *user_name action_name* [*file_argument*]

新用户 (*user_name*) 必须通过下面的机制之一才能对本系统进行显示访问：

- 登录用户的 .Xauthority 文件的可读许可
- **或者** xhost 许可

  例如，下面的两个操作提供了成为超级用户， 并编辑 app-defaults 文件的能力。

```
ACTION AppDefaults
  {
      WINDOW_TYPE         NO_STDIO
      EXEC_STRING          /usr/dt/bin/dtaction -user root \
         EditAppDefaults%Arg _1"文件："%
  }

  ACTION EditAppDefaults
  {
   WINDOW_TYPE         TERMINAL
   EXEC_STRING        /bin/sh -c 'chmod +w%Arg _1%; \
             vi%Arg _1%; chmod -w%Arg _1%'
  }
```

创建本地化的操作

数据类型的查找路径包括与语言相关的位置。桌面利用 LANG 值来确定查找数据类型定义的位置。

本地化操作的位置

本地化操作定义必须放置在沿操作查找路径的合适的语言相关的目录中。

缺省的查找路径为：

- 个人操作：*HomeDirectory* /.dt/types
- 系统范围操作：/etc/dt/appconfig/types/*language*

- 内置操作：/usr/dt/appconfig/types/*language*

▼ 本地化现有的操作

1. **在相应的与语言相关的目录中创建文件 (例如，在 /etc/dt/appconfig/types/japanese 中)。**

2. **将操作定义复制到与语言相关的配置文件。**

   例如，可以复制操作定义从

   app_root/dt/appconfig/types/C/*file*.dt

   复制到

   app_root /dt/appconfig/types/japanese/*newfile*.dt

3. **增加一个 LABEL 字段或修改现有的 LABEL 字段。**

   LABEL **字符串**

   “应用程序管理器”和“文件管理器”利用此标记字符串来标识操作的图标。

4. **在操作定义中本地化任意下面的字段：**

   - 对于本地化图标：ICON
   - 对于本地化“针对项”帮助：DESCRIPTION
   - 对于本地化的提示符：在 EXEC_STRING 中任何带引号的文本

创建 ToolTalk 应用程序的操作

注意：下面的信息只适用于支持 ToolTalk 消息的应用程序。

使用操作类型 TT_MSG 创建一个发送 ToolTalk 消息的操作。

```
ACTION action_name
{
  TYPE         TT_MSG
  ...
}
```

addressing and disposition Fields

- ToolTalk 定址字段总是设置为 TT_PROCEDURE。
- 处置字段的缺省值为静态消息模式中的指定值。

不支持的消息

TT_MSG 类型操作不支持下面项：

- ToolTalk 面向对象的消息
- 消息中的上下文变量

TT_MSG 操作的关键字

表 12–3显示 TT_MSG 操作的关键字及用法。

表 12–3 TT_MSG 操作关键字及用法

关键字	用法
TT_CLASS	定义 ToolTalk 的类消息字段的值
TT_SCOPE	定义 ToolTalk 范围消息字段的值
TT_OPERATION	定义 ToolTalk 操作消息字段的值
TT_FILE	定义 ToolTalk 文件消息字段的值
TT_ARGn_MODE	为 ToolTalk 的第 *n* 个消息变量定义模式属性的值
TT_ARGn_VTYPE	定义 ToolTalk 的第 *n* 个消息变量的 vtype 属性的值
TT_ARGn_VALUE	定义第 *n* 个消息变量的值

第 13 章

手动创建数据类型

有两种创建数据类型定义的方法：

- 使用“创建操作”工具。使用“创建操作”包括在第 11 章。
- 通过手动创建数据类型定义

手动创建数据类型需要对数据库文件进行编辑。

本章描述如何手动创建数据类型定义。

- 有关数据类型的介绍，请参见第 10 章。
- 有关数据类型定义的参考信息，请参见 dtddsfile(4) 手册页。

必须手动创建数据类型的理由

用手动创建数据类型可以使你利用所有内置在数据类型定义语法中的功能

如果要使用这些数据类型的特性，就必须用手动创建数据类型：

- 基于位置 (路径) 的数据分类
- 指定与数据类型关联的， 而不是与“打开”和“打印”的操作的能力
- 相同数据类型的多重名称、模式或内容标准 —例如，基于名为 `*.abc` 或 `*.def` 的文件的数据类型
- 基于链接的数据分类

数据类型定义的组成部分：标准和属性

数据类型定义包括两个分离的数据库定义：

- DATA_ATTRIBUTES 定义。

  DATA_ATTRIBUTES 定义描述了数据类型的名称和此类型文件的外观和性能。
- DATA_CRITERIA 定义。

  DATA_CRITERIA 定义描述了分类标准。每个标准定义指定了该标准所适用的 DATA _ATTRIBUTES 定义。

必须至少有一个 DATA_CRITERIA 定义作为每个 DATA_ATTRIBUTES 定义；而 DATA _ATTRIBUTES 定义可以有多个 DATA_CRITERIA 与其相关联。

例如，可以为 PostScript 文件创建属性定义，此定义描述了 PostScript 文件在"文件管理器"中的外观和性能。然后，可以为 PostScript 数据类型创建两个分离的标准 —一个基于文件名，而另一个基于文件内容。

有关详细信息，请参见第 140 页"数据库 (操作/数据类型) 查找路径"。

手动创建数据类型：一般步骤

本节描述如何创建数据类型的配置文件。

数据类型的配置文件

对包含数据类型定义的配置文件的要求是：

- 文件必须使用命名惯例 *name*.dt
- 文件必须位于数据库查找路径中。缺省的查找路径为：

  个人数据类型—*HomeDirectory*/.dt/types

  系统范围数据类型—/etc/dt/appconfig/types/*language*

  内置数据类型 —/usr/dt/appconfig/types/*language*。不应当使用此目录。

  有关修改数据库查找路径的信息，请参见第 136 页"设置查找路径的值"。

▼ 创建数据类型定义

1. **打开现有的数据库文件或创建一个新的。**

   有关详细信息，请参见前一节第 196 页“数据类型的配置文件”。

2. **为数据类型定义数据属性，使用下面的语法：**

   ```
   DATA_ATTRIBUTESdata_type_name
           {
              ICON                                image_name
             DESCRIPTION                    string
             attribute_field
             attribute_field
             ...
            }
   ```

   其中：

   data_type_name—给定此数据类型的唯一名称。

   image_name—图标文件的文件名或路径。对于文件请使用基本名称。例如，对于图标文件 myimage.m.pm和 myimage.t.pm，请使用 myimage。

   attribute_field—定义数据类型的外观和性能的字段。

   string—字符串。其内容将是数据类型的针对项帮助。

   请参见第 198 页“创建个人操作和数据类型的实例”。

3. **定义数据类型的数据标准，使用下面的语法：**

   ```
   DATA_CRITERIA      criteria_name
         {
         DATA_ATTRIBUTES_NAME                         data_type_name
             criteria_field
             criteria_field
             ...
         }
   ```

   其中：

   criteria_name—此标准定义的唯一名称

   data_type_name—在 DATA_ATTRIBUTES 定义中所使用的名称

   criteria_field—用于定义标准的字段，该标准用于将文件指定给该数据类型

   请参见第 201 页“定义数据类型的数据标准”。

4. **保存数据库文件。**

5. **为数据类型创建图标。**

   有关详细信息，请参见第 199 页“指定用于数据类型的图标图像”。

6. **如果有必要，创建在属性定义的 ACTIONS 字段列出的操作。**

7. **双击 Desktop_Controls 应用程序组中的“重新装入操作”，以便重新装入数据库。**

创建个人操作和数据类型的实例

假设系统中包含名为 `xgif` 的应用程序，它是用来显示 GIF 图片的。通常通过执行下面的命令来运行该程序：

`xgif` *filename*

需要以几种方式来显示 GIF 图片：

- 通过双击 GIF 数据文件
- 通过选择数据文件，并在"选中"菜单中选择应用程序

1. **打开一个新文件** *HomeDirectory*/`.dt/types/GifViewer.dt`**，以便编辑。**

2. **键入数据类型定义：**

```
DATA_ATTRIBUTES Gif
 {
     DESCRIPTION                Gif image file.
     ICON                        GifIcon
     ACTIONS                    View
 }

 DATA_CRITERIA Gif_Criteria
 {
     DATA_ATTRIBUTES_NAME            Gif
     NAME_PATTERN                        *.gif
 }
```

3. **键入 GifViewer 操作的操作定义：**

```
ACTION GifViewer
     {
         EXEC_STRING                    xgif %(File)Arg_1"Gif file to view:"
         WINDOW_TYPE                    NO_STDIO
         DESCRIPTION                    Double-click or drop a file to \
                                         start the Gif viewer.
     }
```

因为定义不包括 `ICON` 字段，所以该操作使用系统缺省图标。

4. **键入下面的映射操作，将 GifViewer 操作连接到在数据类型定义中列出的"查看"操作。使用** `ARG_TYPE` **字段将此查看操作限定到 Gif 类型文件。**

```
ACTION View
     {
         ARG_TYPE                Gif
         TYPE                    MAP
         MAP_ACTION            GifViewer
     }
```

5. **保存文件。**

6. **双击在 Desktop_Controls 应用程序组中的"重新装入操作"，以重读数据库。**

定义数据类型的数据属性

DATA_ATTRIBUTES 定义定义了数据类型的外观和性能。它指定了数据类型的名称，并提供了指定下面的各项的能力：

- “文件管理器”图标 (ICON 字段)
- “选中”菜单的双击性能和内容 (ACTIONS 字段)
- 数据类型的针对项帮助 (DESCRIPTION 字段)

指定用于数据类型的图标图像

使用 ICON 字段指定用于“文件管理器”的图标。如果未指定图标图像，“文件管理器”只显示一个标号。

ICON 字段的值可以是：

- 基本文件名。

  基本文件名是包含图标图像文件的名称，减去文件名用以表示大小 (m 和 t) 和图像类型 (bm 和 pm) 的后缀。例如，如果文件名为 GameIcon.m.pm 和 GameIcon.t.pm，则使用 GameIcon。

  如果使用基本文件名，图标文件必须放置在图标查找路径的目录中：

  - 个人图标：*HomeDirectory* /.dt/icons
  - 系统范围图标：/etc/dt/appconfig/icons/*language*
- 图标文件的绝对路径，包含全文件名。

  只有当图标文件并不位于图标查找路径下，才应使用绝对路径。例如，如果图标文件 GameIcon.m.pm 放在目录 /doc/projects (不在图标查找路径) 下，ICON 字段的值则为 /doc/projects/GameIcon.m.pm。

  表 13–1 列出将要创建的图标的大小以及相应文件名。

表 13–1 数据类型图标的图标名称和大小

像素大小	位图名称	像素图名称
32 乘 32	**名称** .m.bm	**名称** .m.pm
16 乘 16	**名称** .t.bm	**名称** .t.pm

将数据类型与操作相关联

数据类型与操作有两种关联方法：

- 位于 DATA_ATTRIBUTES 定义中的 ACTIONS 字段列出了将出现在“文件管理器”的“选中”菜单中的操作。列表中的第一个操作是缺省 (双击) 操作。
- 使用操作定义的 ARG_TYPE 字段，可将操作限定为指定数据类型。

  例如，下面的数据类型定义为特殊的“自述”文件创建了一种数据类型，而这些文件由系统管理员创建，使用的命名惯例是 *.rm。

```
DATA_ATTRIBUTES SysReadmeFile
 {
     ICON                        SysReadMe
     ACTIONS                   Open，Respond
 }
 DATA_CRITERIA SysReadmeFileCriteria
 {
     NAME_PATTERN                        *.rm
     DATA_ATTRIBUTES_NAME             SysReadmeFile
 }
```

  在下面为文件定义了特殊的“响应”操作。它在“文本编辑器”中打开文件的一个可写副本。当保存文件并退出“文本编辑器”后，文件被寄给系统管理员 (邮寄地址为 sysadmin@utd)。

```
ACTION Respond
 {
     ARG_TYPE                SysReadmeFile
     EXEC_STRING            /bin/sh -c 'cp %Arg_1% $HOME/readme.temp;\
                             chmod +w $HOME/readme.temp;             \
                             dtpad $HOME/readme.temp;                \
                             cat $HOME/readme.temp |                 \
                             /usr/bin/mailx sysadmin@utd;            \
                             rm $HOME/readme.temp'
     WINDOW_TYPE            NO_STDIO
 }
```

根据数据类型隐藏文件

如果文件是不可见数据类型，就决不会出现在“文件管理器”中。

使用位于 DATA_ATTRIBUTES 定义中的 PROPERTIES 字段来指定此类要隐含的对象：

```
PROPERTIES                不可见
```

指定当文件被操作时的性能

表 13–2 显示主要由应用程序程序程序员使用的 DATA_ATTRIBUTES 字段。它们指定了在用户执行各种桌面操作时，文件所表现出的特性。

有关详细信息，请参见《 **公共桌面环境程序员指南** 》，这是开发者环境文档的一部分。

表 13–2 DATA_ATTRIBUTES 字段和描述

字段	描述
MOVE_TO_ACTION	用于类似目录这样的容器。指定当文件被移到该数据类型的容器中时，所要执行的操作。
COPY_TO_ACTION	用于类似目录这样的容器。指定当文件被复制该数据类型的容器中时，所要执行的操作。
LINK_TO_ACTION	指定当文件被链接到该数据类型的文件时，所要执行的操作。
IS_TEXT	指定该数据类型的文件含有可在文本框 中显示的文本。
MEDIA	指定相应的 ToolTalk 媒体类型。
MIME_TYPE	指定相应的 MIME 类型。
X400_TYPE	指定相应的 X400 类型。

定义数据类型的数据标准

DATA_CRITERIA 定义定义了将对象类型指定给文件或目录所采用的标准。

可使用在对象分类表 13–3 中显示的标准。

表 13–3 DATA_CRITERIA 标准和描述

标准	描述
文件名	文件名必须与指定模式相匹配。使用 NAME_PATTERN 字段。
文件位置	路径必须与指定模式相匹配。使用 PATH_PATTERN 字段。
文件内容	文件内容的指定部分必须与指定数据相匹配。使用 CONTENT 字段。
文件模式	文件必须拥有指定的许可 (读、写、执行、目录)。使用 MODE 字段。
符号链接	分类是以对象所链接的文件为基础的。

对于一种数据类型，可以使用多个标准。然而，不能在同一个数据类型中使用 NAME _PATTERN 和 PATH_PATTERN 标准。

基于名称的数据类型

使用 NAME_PATTERN 字段指定命名要求。字段值可以包含下面的通配符：

?—匹配任何单个字符

*—匹配任何字符序列 (包括空字符串)

[*cc* …]—匹配括在括号中的任何字符 (*c*)

[*c*–*c*]—匹配字符的范围为 *c* 到 *c* 的任何字符

实例

- 下面的数据类型定义创建了基于文件名的数据类型。文件名必须以 QS 开头，并以 .doc 结尾。

```
DATA_ATTRIBUTES QS_Doc
 {
    DESCRIPTION        This file contains a document for the QS \
                    project.
    ICON            Word_Doc
    ACTIONS            Open
 }

 DATA_CRITERIA QS_Doc_Criteria
 {
    NAME_PATTERN            QS*.doc
    DATA_ATTRIBUTES_NAME     QS_Doc
 }
```

- 下面的定义为名为 Demo_*n* 其中， *n* 为 0 到 9。

```
DATA_ATTRIBUTES Demo_directory
 {
    DESCRIPTION        This is a directory. Double-click to open it.
    ICON            Demo
    ACTIONS            OpenInPlace，OpenNewView
 }

 DATA_CRITERIA Demo_directory_criteria
 {
    NAME_PATTERN                Demo_[0-9]
    MODE                         d
    DATA_ATTRIBUTES_NAME        Demo_directory
 }
```

基于位置的数据类型

使用 PATH_PATTERN 字段指定路径。可以使用与 NAME_PATTERN 相同的通配符。

例如，下面的数据类型使用了基于路径的标准。

```
DATA_ATTRIBUTES Project_Graphics
 {
    DESCRIPTION        Graphics file for the QS project. Double-click the \
                    icon to see the graphic.
```

```
    ICON            QSgraphics
}
DATA_CRITERIA Project_Graphics_Criteria
{
    DATA_ATTRIBUTES_NAME           Project_Graphics
    PATH_PATTERN                       */projects/QS/graphics/*
}
```

基于名称和位置的数据类型

要创建基于文件名和位置的数据类型，请在 PATH_PATTERN 值中包括此名称。在同一个标准定义中，不能同时使用 NAME_PATTERN 和 PATH_PATTERN。

实例

- 下面定义的 QS_Source_Files 数据类型适用于所有名为 app*n* .c 的文件，其中*n* = 1 到 9， 这些文件位于 */projects/QS 的子目录中。

```
DATA_ATTRIBUTES QS_Source_Files
 {
    ...
 }

 DATA_CRITERIA QS_Source_Files_Criteria
 {
     PATH_PATTERN                       */projects/QS/*/app[1-9].c
     DATA_ATTRIBUTES_NAME            QS_Source_Files
 }
```

- 下面的数据类型适用于所有在目录 /doc/project1 中的，名为 ch*nn*.*xxx* 的文件，其中， *n* 为 0 到 9，*xxx* 为任何三个字符的文件名后缀。

```
DATA_ATTRIBUTES ChapterFiles
 {
     DESCRIPTION            Chapter file for the project document.
     ICON                    chapter
     ACTIONS                Edit，Print
 }

 DATA_CRITERIA Chapter_Criteria
 {
     PATH_PATTERN                       /doc/project1/ch[0-9][0-9
].???
      DATA_ATTRIBUTES_NAME            ChapterFiles
 }
```

使用文件模式作为分类标准

使用 MODE 字段指定必需的许可。

模式标准通常与基于名称、基于位置或基于内容的数据分类组合使用。从而可以将数据类型限定在文件或目录上，或指定所需的读、写和执行许可。

MODE 字段可包含逻辑运算符 (表 13–4) 和字符 (表 13–5)。

表 13–4 MODE 字段逻辑运算符和描述

运算符	描述
!	逻辑运算符 NOT
&	逻辑运算符 AND
|	逻辑 OR

表 13–5 MODE 字段字符和描述

字符	描述
f	该数据类型只适用于文件
d	该数据类型只适用于目录
r	该文件对于任何用户都是可读的
w	该文件对于任何用户都是可写的
x	该文件对于任何用户都是可执行的
l	该文件是链接

某一具体模式的缺省状态与模式无关。

实例

- 下面的模式字段对数据类型有如下限定：

  f&!w—只读文件

  !w—只读文件和目录

  f&x—可执行文件

  f&r&x—既可写又可执行的文件

  x|!w—可执行或只读文件

- 下面的数据类型定义为只读、不可执行文件创建了数据类型，这些文件的文件名符合命名惯例 *.doc。这假定已为数据类型定义了“查看”操作。

```
    DATA_ATTRIBUTES ReadOnlyDocument
     {
         ICON                        read_only
         DESCRIPTION                This document is not writable.
 Double- \
```

```
                                        clicking runs your editor with a \
                                        read-only copy of the file.
    ACTIONS                            View
}

DATA_CRITERIA ReadOnlyDocument_Criteria
{
    NAME_PATTERN                                *.doc
    MODE                                            !d&!x&!w
    DATA_ATTRIBUTES_NAME              ReadOnlyDocument
}
```

基于内容的数据分类

使用 CONTENT 字段来指定基于文件内容的数据分类。基于内容的数据分类可与基于名称或基于位置的数据分类组合使用。

分类可基于文件的字符串或数字内容。文件的第一字节的编号为 0 。

- 对于文件的字符串内容，请使用语法：

  CONTENT *starting_byte* string*string*

- 对于文件的数字内容，请使用语法：

  CONTENT *starting_byte* byte*number*
  CONTENT*starting_byte* short*number*
  CONTENT*starting_byte* long*number*

- 对于目录的内容，请使用语法：

  CONTENT 0 filename "*file_name*"

使用标准 C 的八进制 (前导 o) 和十六进制 (前导 oX) 数字的记数法。

注意 : 使用基于内容的数据分类将导致系统性能减慢。只要有可能，就换用基于名称和位置的分类。

例如，下面的数据类型，Writable_Wingz，适用于所有具有写许可的文件，这些文件在文件的开始处含有字符串 WNGZ。

```
DATA_ATTRIBUTES Writable_Wingz
{
     ...
 }

 DATA_CRITERIA Writable_Wingz_Criteria
 {
     CONTENT                              0 string WNGZ
     MODE                                  w&!d
```

```
    DATA_ATTRIBUTES_NAME    Writable_Wingz
}
```

▼ 用几种独立标准创建数据类型

可用几个独立的标准来创建数据类型—就是说，如果文件符合标准中的任一个 (或两者)，就为其指定数据类型。

1. **为数据类型创建 `DATA_ATTRIBUTES` 定义。**
2. **为每个标准创建 `DATA_CRITERIA` 定义。**

   使用 DATA_ATTRIBUTES_NAME 字段将每个标准连接到同一个 DATA_ATTRIBUTES 定义。

   例如，下面的定义创建了 Mif 数据类型。分类是基于名称或内容的。

```
DATA_ATTRIBUTES Mif
{
     ICON                                Frame
     ACTION_LIST                        Open，Print
 }

 DATA_CRITERIA Mif_Name_Criteria
 {
     DATA_ATTRIBUTES_NAME                Mif
     NAME_PATTERN                            *.mif
 }

 DATA_CRITERIA Mif_Content_Criteria
 {
     DATA_ATTRIBUTES_NAME        Mif
     CONTENT                            1 string MIFFile
 }
```

创建本地化的数据类型

数据类型的查找路径包含与语言相关的位置。桌面使用 LANG 的值来确定查找数据类型定义的位置。

本地化数据类型的位置

本地化的数据类型定义必须放置沿操作查找路径的与适当的语言相关的正确目录中。

缺省的查找路径为：

- 个人操作：*HomeDirectory*/`.dt/types`

- 系统范围操作：`/etc/dt/appconfig/types/`*language*
- 内置操作：`/usr/dt/appconfig/types/`*language*

▼ 本地化数据类型

1. **在相应的与语言相关的目录中 (例如，在** `/etc/dt/appconfig/types/japanese` **中) 创建文件。**
2. **将数据类型定义复制到与语言相关的配置文件中。**
3. **本地化数据类型定义中的一个或多个字段。**

第 14 章

为桌面创建图标

桌面上的图标与下面的对象相关联：

- 在“文件管理器”和“应用程序管理器”中的操作文件和数据类型
- “前面板”控制
- 已最小化的应用程序窗口
- 如调色板和工具栏这样的应用程序所使用的图形
- 工作区的背景

注意：开发环境文档包括桌面图标的附加信息。请参见第四章，*Common Desktop Environment: Style Guide and Certification Checklist*中的“可视化设计”。

图标图像文件

为使桌面能够使用图标图像，此图标图像文件必须：

- 具有适当的格式。
- 符合适当的文件命名惯例。
- 符合桌面大小惯例。
- 在图标查找路径的目录中。
- 使用适当的语法，用桌面结构调用。例如，如果在“前面板”上创建一个新控制，则使用“前面板”定义中的 `ICON` 字段来指定用于该控制的图标图像。

图标文件格式

对于彩色显示，应使用 X 像素图 (XPM) 格式的图标文件，这种文件通常以 .pm 作后缀。或者，使用 X 位图 (XBM) 格式的文件，这种文件通常以 .bm 作后缀。如果在像素图文件中使用了透明度，则掩码文件 (_m.bm) 与 .bm 文件同时生成。有关桌面如何查找这些文件的详细信息，请参见第 142 页"图标查找路径"。

图标文件名

每个图标和背景图像均存储为独立的文件。通常情况下，图标由其文件名的基本部分指定。例如，在文件以下面的名称存储时，该图标通常以名称 mail 来引用：

/usr/dt/appconfig/icons/*language*/mail.l.pm

增加文件后缀的文件命名惯例有助于按照大小和类型对图标进行归类。 桌面组件的图标名称包括在下面的一般格式之中：

基本名称 .大小 .格式

或

基本名称 .格式

其中：

基本名称—引用此图像时所使用的图像基本名称

大小 —表明文件大小的字符： l (大) m (中) s (小) t (特别小)

格式 —文件格式： pm (像素图文件) bm (位图文件)

图标大小惯例

表 14–1显示桌面图标的建议像素尺寸。

表 14–1 图标大小及文件名

图标大小	位图名称	像素图名称
16 乘 16 (特别小)	*name* .t.bm	*name* .t.pm
24 乘 24 (小)	*name* .s.bm	*name* .s.pm
32 乘 32 (中)	*name* .m.bm	*name* .m.pm
48 乘 48 (大)	*name* .l.bm	*name* .l.pm

表 14–2 显示桌面组件所使用的图标大小。在某些情况下，所用图标的大小取决于显示器的分辨率。

表 14–2 桌面组件及其图标大小

桌面组件	高分辨率	中分辨率	低分辨率
“文件管理器”和“应用程序管理器”(通过名称和图标查看)	中	中	中
“文件管理器”和“应用程序管理器”(通过名称和小图标查看)	特别小	特别小	特别小
“主前面板”控制	大	大	中
“前面板”的子面板	中	中	特别小
“前面板”切换控制	小	小	特别小
最小化窗口	大	大	中

例如，如果为某一数据类型指定名为 mail 的图标，并且为彩色显示，而且设置“文件管理器”的首选项为小图标，则所用的此图标图像文件的文件名为 `mail.t.pm`。

图标查找路径

桌面通过搜索目录列表的文件来查找图标文件或图像。目录列表也称为**图标查找路径**，是由几个环境变量的值决定的。使用哪些变量和如何将它们放在一起来创建图标查找路径，将在第 142 页“图标查找路径”中讨论。

缺省的查找路径为：

- 内置图标：`/usr/dt/appconfig/icons/`*language*
- 系统范围图标： `/etc/dt/appconfig/icons/`*language*
- 个人图标： *HomeDirectory* `/.dt/icons`

通过网络访问图标

桌面可访问远程系统中的图标。有关创建图标服务器的详细信息，请参见第 124 页“配置数据库、图标和帮助服务”。

图标关联

为了能更快捷地识别对象，可将图标与下面的对象相关联：

- 操作和数据类型
- “前面板”和子面板中的控制

■ 已最小化的应用程序窗口

指定图标文件

对用于操作、数据类型的图标以及在“前面板”或子面板中使用的图标，仅指定图标的基本名称 (没有后缀)。其正确的后缀是根据显示器分辨率、所支持的颜色数和“文件管理器”的查看选项 (例如以小图标方式) 自动添加的。

覆盖查找路径，需要提供图标文件的完整路径和文件名。

▼ 将图标与操作或数据类型相关联

1. 使用 `ICON` 字段来指定图标。

如果按照恰当的图标文件命名惯例，只需指定图标的基本名称。将根据显示器的分辨率和所支持的颜色数显示正确的图标。

2. 创建下面的图标尺寸：

- 操作：大、中、特别小
- 数据类型：中、特别小

操作定义实例

下例是启动 Island Paint 绘画工具的操作定义。图标 `Ipaint.l` 和 `Ipaint.s`与 此操作相关联。

```
ACTION    IslandPaintOpenDoc
{
     WINDOW_TYPE               NO-STDIO
     ICON                       Ipaint
     EXEC_STRING           /usr/bin/IslandPaint %Arg_1"要打开的文件："%
 }
```

如果使用彩色图标，桌面系统查找实际的图标文件时，首先附加 .pm 后缀；否则 (或者没有找到与 .pm 后缀相匹配的文件时)，桌面系统将附加 .bm 后缀。

数据类型定义实例

下面的数据类型的定义使图标 `comprsd.l` 和 `comprsd.s` 与压缩文件相关联：

```
DATA_ATTRIBUTES  COMPRESSED
{

        ICON                    comprsd
     ACTIONS               Uncompress
     DESCRIPTION           A COMPRESSED file has been compressed by the \
```

```
                                    'compress' command to take up less space.
}
```

▼ 显示在前面板控制中的图标

1. **使用 `ICON` 字段来指定图像名称。**

   如果控制监视一个文件（`MONITOR_TYPE` 设置为邮件或文件)， 当检测到更改时，则使用 `ALTERNATE_ICON` 字段指定使用的图标。

   也可以提供按钮和拉放区控制的动画。

2. **创建下面的图标尺寸：**

   - “前面板”和子面板：大、中和特别小
   - 工作区切换：小

实例

当名为 `report` 的文件存放于 `/doc/ftp/pub/` 目录中时，下面的控制会更改外观。当此文件不存在时，将显示图标文件 `NoReport.pm`；反之，则将显示图标文件 `Report.pm`。

```
CONTROL MonitorReport
 {
     CONTAINER_NAME        container_name

   TYPE                                ICON
   MONITOR_TYPE                    file
     FILE_NAME                         /doc/ftp/pub/report
     ICON                                 NoReport
     ALTERNATE_ICON                    Report
 }
```

▼ 将图标与应用程序窗口相关联

1. **为“工作区管理器”设置 iconImage 资源如下：**

   `Dtwm*`*clientname*`*iconImage:` *icon_file_name*

   要确定 *clientname* 的正确值，需打开“应用程序管理器”，并在 Desktop_Tools 应用程序组中双击“窗口属性”。选中窗口后，此窗口的属性被列出。其中 WM_CLASS 属性在引号中显示窗口的类名。

   有关设置资源的详细信息，请参见第 257 页“设置应用程序资源”。

2. **在“工作区”菜单中选择“重新启动工作区管理器”。**

   验证“工作区管理器”已经识别出图标，请将要修改图标的窗口最小化。

注意：有些应用程序不允许覆盖它们的缺省窗口图标。

▼ 将文件管理器作为图标浏览器使用

1. **将文件** /usr/dt/examples/*language* /IconBrowse.dt **复制到** *HomeDirectory* /.dt/types/Iconbrowse.dt**目录中。**

2. **打开"应用程序管理器"，并在 Desktop_Tools 应用程序组内双击"重新装入操作"。**

   进入含有图标 (.bm 和 .pm 文件) 的目录时，每个图标即显示在其名称旁边。例如，如果进入目录 /usr/dt/appconfig/icons/*language*， 将能看到诸多桌面图标。

注意：在内存较小的系统中启用图标浏览会使"文件管理器"显示目录时速度更慢些。在缺省的配置下不显示大于 256 乘 256 的图像。

要禁用图标浏览：

1. **删除个人的** IconBrowse.dt **文件的副本。**

2. **打开"应用程序管理器"，并在 Desktop_Tools 应用程序组内双击"重新装入操作"。**

图标设计建议

在相关的图标间使用公共的主题。例如，如果为应用程序设计图标，应有意识地使设计的应用程序图标与其数据文件的图标相类似。

请确保所设计的任何彩色图标的黑白两色版本为系统所接受。如果此图标是显示在单色的或灰度的显示器上 (或者显示器没有足够的颜色可用)，图标则自动以黑白两色版本的形式显示。

要保留系统的颜色使用情况，应将图标颜色的使用限制在桌面提供的颜色之内。(由"图标编辑器"所创建的图标将只使用桌面提供的颜色)

有关桌面组件所使用的尺寸，请参见表 14–1。

颜色的使用

桌面图标使用 22 种颜色的调色板，其中包括：

- 八种静态灰色

- 八种静态颜色：红色、蓝色、绿色、青色、洋红、黄色、黑色 和白色。
- 六种动态颜色：前景色、背景色、顶部阴影、底部阴影、选中和透明色

  此调色板在不占用其他应用程序所需的颜色资源的情况下，创建吸引人的、便于阅读的图标。桌面提供的大多数图标使用加重色调的灰色。

  因为图标后面的颜色可以透过图标，所以透明色有助于创建具有非矩形错觉的图标。

第 15 章

高级前面板定制

用户可以使用其弹出式菜单和在子面板中的“安装图标”控制，定制“前面板”。

本章介绍通过创建和编辑配置文件来定制“前面板”的内容。

- 有关“前面板”控制和配置的参考信息，请参见 `dtfpfile`(4X) 手册页。
- 有关“工作区管理器”的参考信息，请参见 `dtwm`(1) 和 `dtwmrc`(4) 手册页。

前面板配置文件

“前面板”定义在配置文件的数据库中。

配置文件提供定制“前面板”的方法。某些修改只有通过编辑配置文件才可完成。 其中包括：

- 向“主面板”添加新的控制位置。
- 添加特殊的控制类型，例如，客户机窗口。
- 更改某种缺省性能—例如，“前面板”控制是否响应单击或双击。

要在配置面板时提供最大的灵活性，这些文件可以是个人的、系统范围的或位于其他系统上。

"前面板"由"工作区管理器"创建和管理。

缺省的前面板配置文件

缺省的"前面板"定义在"前面板"配置文件 /usr/dt/appconfig/types/*language* /dtwm.fp 中。

请勿更改此文件。

前面板配置文件的查找路径

"前面板"定义可以在任意数目的位于本地或在远程系统上的文件中分发。

用于定义"前面板"的文件必须符合这些要求：

- 文件名必须以 .fp 结尾；例如，mail.fp。
- 文件必须位于操作数据库查找路径中。

缺省的操作数据库查找路径包含这些目录，按下面的次序搜索：

个人的定制—*HomeDirectory*/.dt/types

系统范围的定制—/etc/dt/appconfig/types/*language*

内置的面板和控制—/usr/dt/appconfig/types/*language*

附加的目录，*HomeDirectory*/.dt/types/fp_dynamic，用于使用用户接口进行的个人定制。 请勿使用此目录进行手动定制。

操作数据库查找路径可能包含添加到配置联网系统的附加目录。尤其是，在配置系统访问应用程序服务器时，添加附加远程位置。有关详细信息，请参见第 140 页"数据库 (操作/数据类型) 查找路径"。

如何装配前面板：优先级规则

"前面板"从所有位于操作数据库的查找路径的配置文件装配。

如果定义中的组件之间有冲突，由优先级规则决定使用哪一个定义。在下面的情况下，两个组件冲突：

- 具有相同的控制名称，CONTAINER_NAME 和 CONTAINER_TYPE。
- 竞争相同的位置 (由于具有不同名称，但是相同的 CONTAINER_NAME, CONTAINER _TYPE 和 POSITION_HINTS)。

"前面板"使用下面的优先级规则：

- 如果组件的控制名、容器名和类型相同，将使用首先读取的组件。

  例如，如果系统范围控制和内置控制都包含这些字段，但其他方面不同：

  ```
  CONTROL TextEditor
   {
       CONTAINER_TYPE                 BOX
       CONTAINER_NAME              Top
       ...
   }
  ```

  那么，系统范围控制具有优先权。

- 如果两个组件竞争同一位置，那么，按它们的读取次序放置。

  例如，如果用户创建新的“主面板”的个人控制 (CONTAINER_TYPE BOX 和 CONTAINER_NAME Top) 并指定它为 POSITION_HINTS 5，个人控制将碰撞内置控制和所有其他控制，使它们向右移动一个位置而具有了更高位置号。

注意：通过创建新系统范围版本或个人版本来修改控制时，新的控制定义必须指定具有相同的控制名称 CONTAINER_NAME 和 CONTAINER_TYPE。否则，新的控制将出现在现有控制之外。

动态创建的前面板文件

用户使用“安装图标”控制和弹出式菜单定制“前面板”时，文件被写到目录 *HomeDirectory* /.dt/types/fp_dynamic 中。

“前面板”创建一个附加文件 *HomeDirectory* /.dt/sessions/dtwmfp.session，它用于保存和恢复每个会话的定制的“前面板”状态。

管理用户接口定制

用户可使用“前面板”控制的弹出式菜单和“安装图标”控制来广泛地定制“前面板”。

此节描述如何进行：

- 防止某些个人的定制。例如，您可能想使用户无法删除控制。
- 撤销个人的定制。例如，用户可能请求恢复一个意外删除的控制。

▼ 防止个人的定制

1. **如果控制是内置控制，将它的定义从** /usr/dt/appconfig/types/*language*/dtwm.fp **复制到** /etc/dt/appconfig/types/*language*/*name*.fp 中。

2. **在控制的定义中添加下行：**

```
LOCKED   True
```

▼ 恢复被删除的控制或子面板

在 Desktop_Tools 应用程序组中"恢复前面板"操作删除所有通过用户接口进行的"前面板"定制。 用户可使用此操作删除其所有通过"前面板"弹出式菜单进行的个人定制。

使用下面的过程恢复个人控制。

- **在 *HomeDirectory* `/.dt/types/fp_dynamic`目录中，删除用户删除控制时创建的文件。控制将与删除的原始控制同名。**

例如，如果用户删除了"图标编辑器"控制，在 `fp_dynamic` 目录中的一个文件将包含：

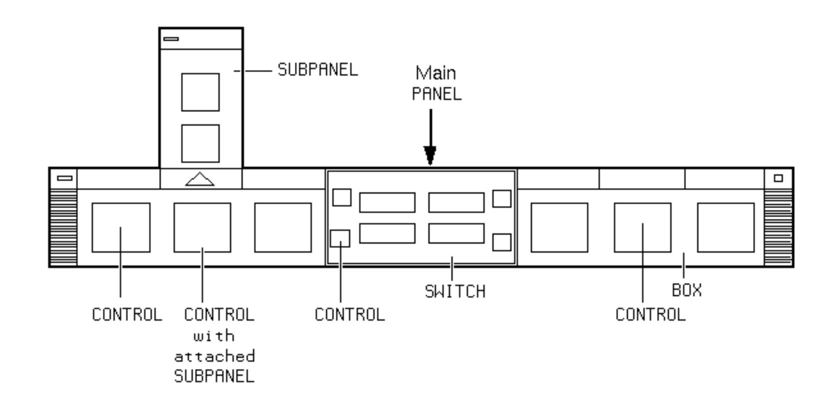
```
    CONTROL IconEditor
{
        ...
       DELETE   True
    }
```

用户删除子面板时，将为子面板和子面板中的每个控制创建单独的动态文件。

前面板定义的结构

"前面板"通过装配其组件定义而创建。每个组件都有必需的语法。该语法定义了组件在"前面板"中的放置位置、其外观和性能。

"前面板"组件



图形 15–1 前面板组件

“前面板”从外向内装配：

- PANEL 是整个“前面板”的顶层容器或父容器。
- PANEL 是一个或多个 BOX 的容器。
- 一个 BOX 是一个或多个 CONTROL 的容器。

  有两种特殊类型的容器：

- SUBPANEL 和特殊控制相关联 (此控制是子面板的容器)。子面板从与它相关联的控制“slide up”。
- SWITCH 包含用于更改工作区加上附加控制的按钮。

前面板定义的一般语法

“前面板”中的每个组件是使用下面的语法分别定义的：

```
COMPONENT  name{
    KEYWORD           value
   KEYWORD           value
    . . .
 }
```

一些关键字是必需的，另一些只供选用。有关详细信息，请参见 dtfpfile(4X) 的手册页。

PANEL 定义

PANEL 是顶层组件。它的定义包括：

- “前面板”名称
- 描述整个“前面板”一般外观和性能的字段

```
PANEL front_panel_name
{
    KEYWORD                    value
    KEYWORD                    value
    ...
}
```

front_panel_name 是“前面板”的唯一名称。缺省名称为“FrontPanel”。

BOX 定义

BOX 定义描述：

- BOX name
- 框所在的 PANEL (CONTAINER_NAME)
- PANEL 中框的位置 (POSITION_HINTS)
- 描述用于整个框的外观和性能的字段

```
BOX box_name
{
    CONTAINER_NAME             front_panel_name
    POSITION_HINTS             position
    KEYWORD                    value
    KEYWORD                    value
    ...
}
```

CONTROL 定义

CONTROL 定义描述了：

- CONTROL 名称
- 控制是否在框、子面板或切换中 (CONTAINER_TYPE)
- 控制所在的框、子面板或切换中 (CONTAINER_NAME)
- 在 BOX 中 CONTROL 的位置 (POSITION_HINTS)。
- 描述控制的外观和性能的字段

```
CONTROL control_name
{
    CONTAINER_TYPE         BOX 或 SUBPANEL 或 SWITCH
    CONTAINER_NAME         box_name 或 subpanel_name 或 switch_name
    TYPE                   control_type
    POSITION_HINTS         position
    KEYWORD                value
    KEYWORD                value
...
}
```

SUBPANEL 定义

SUBPANEL 定义描述了：

- SUBPANEL 名称
- 附加子面板的控制的名称 (CONTAINER_NAME)
- 描述子面板的外观和性能的字段

```
SUBPANEL subpanel_name
{
    CONTAINER_NAME              control_name
    KEYWORD          value
    KEYWORD                     value
    ...
}
```

SWITCH 定义

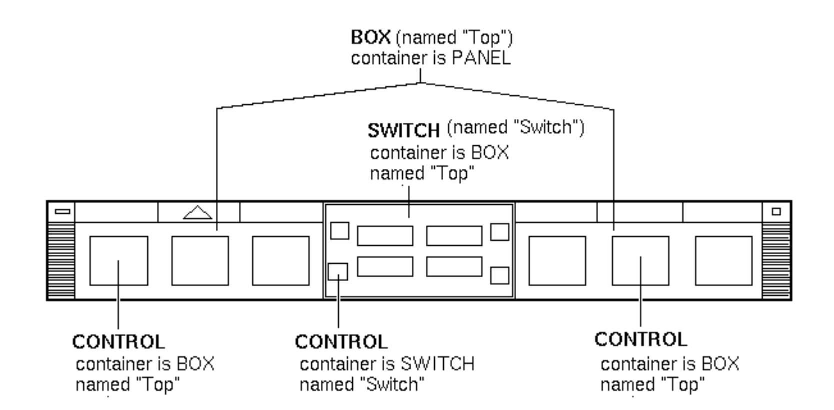
SWITCH 定义描述：

- SWITCH 名称
- SWITCH 所在的 BOX (CONTAINER_NAME)
- 在 BOX 中 SWITCH 的位置 (POSITION_HINTS)
- 描述 SWITCH 的外观和性能的字段

```
SWITCH switch_name
{
    CONTAINER_NAME              box_name
    POSITION_HINTS              position
    KEYWORD                     value
    KEYWORD          value
    ...
}
```

修改主面板

"主面板"是"前面板"窗口，不包括子面板。



图形 15–2 “主面板”容器

可进行的修改包括：

- 添加或删除控制
- 交换控制的位置

▼ 向主面板添加控制

1. **创建“前面板”配置文件：**
   - 系统范围：/etc/dt/appconfig/types/*language* /*.fp
   - 个人：*HomeDirectory* /.dt/types/*.fp

2. **在文件中定义控制。**

   使用 CONTAINER_NAME 和 CONTAINER_TYPE 字段指定控制的容器：

```
CONTAINER_NAME   Top
 CONTAINER_TYPE   BOX
```

   使用 POSITION_HINTS 来指定控制从左到右的位置。因为定制具有高于内置控制的优先级，新的控制将“碰撞”现有控制，使其向右移动一个位置。

3. **保存配置文件。**

4. **创建“前面板”控制的图标。**

   请参见第 227 页“指定控制所用的图标”。

5. **从“工作区”菜单中选择“重启工作区管理器”。**

   例如，在文件 /etc/dt/appconfig/types/*language*/audio.fp 中的下面控制定义在“时钟”和“日历”之间插入音频应用程序控制。

```
CONTROL AudioApplication
  {
   TYPE                icon
```

```
  CONTAINER_NAME   Top
  CONTAINER_TYPE   BOX
  ICON             AudioApp
  POSITION_HINTS   2
  PUSH_ACTION      StartAudioApplication
  PUSH_RECALL      true
 }
```

▼ 删除控制

1. **创建"前面板"配置文件：**
   - 系统范围：/etc/dt/appconfig/types/*language*/*name* .fp
   - 个人：*HomeDirectory* /.dt/types/*name* .fp
2. **将想要删除的控制的定义复制到新文件中。**

   如果控制是内置的，则其定义在 /usr/dt/appconfig/types/*language*/dtwm.fp 中。

   不必复制整个定义。然而，复制部分必须包括 CONTAINER_NAME 和 CONTAINER _TYPE 字段。
3. **将 DELETE 字段添加到定义中：**

   ```
   DELETE             True
   ```
4. **保存配置文件**
5. **从"工作区"菜单中选择"重启工作区管理器"。**

   例如，在文件 /etc/dt/appconfig/types/*language*/TrashCan.fp 中放置的下面控制定义从"前面板"中删除"垃圾篓" 控制。

   ```
   CONTROL Trash
     {
      CONTAINER_NAME  Top
      CONTAINER_TYPE  BOX
      DELETE          True
     }
   ```

▼ 修改控制

需要修改控制定义时—例如，更改它的图标图像，请使用此过程。

1. **从 /usr/dt/appconfig/types/*language*/dtwm.fp 复制全部控制定义：**
   - 系统范围：/etc/dt/appconfig/types/*language*/*name*.fp
   - 个人：*HomeDirectory* /.dt/types/*name*.fp.
2. **编辑想要更改的字段。也可添加附加的字段。**

3. **保存文件**

4. **从“工作区”菜单中选择“重启工作区管理器”。**

▼ 交换控制的位置

1. **复制要将位置从** /usr/dt/appconfig/types/*language*/dtwm.fp **变为下面位置的控制的控制定义：**

   - 系统范围：/etc/dt/appconfig/types/*language*/*name* .fp
   - 个人：*HomeDirectory* /.dt/types/*name*.fp.

   必须复制每个要移动的控制的整个控制定义。

2. **交换控制定义** POSITION_HINTS **字段的值。**

3. **保存文件**

4. **从“工作区”菜单中选择“重启工作区管理器”。**

   例如，放置在文件 /etc/dt/appconfig/types/C/MailHelp.fp 中的下面的定义交换了“邮件”和“帮助管理器”控制的位置，并锁定这些控制，以防个人更改。

```
CONTROL Mail
 {
     POSITION_HINTS                    12
     LOCKED                            True
     . . .the rest of the control definition
 }


CONTROL Help
 {
     POSITION_HINTS                    5
     LOCKED                            True
     . . .the rest of the control definition
 }
```

▼ 替换前面板控制

● **创建另一个控制定义，用相同的：**

- *control_name*
- CONTAINER_NAME 值

  例如，下两个控制定义在两个不同的配置文件中。它们有相同的控制名和容器名，因此认为它们是相同的控制。

- /etc/dt/appconfig/types/C/SysControls.fp 中的定义：

```
Control ImportantApplication
 {
```

```
        CONTAINER_NAME                    Top
        CONTAINER_TYPE                    BOX
        POSITION_HINTS                    2
  ...
}
```

- *HomeDirectory* /.dt/types/MyControls.fp 中的定义：

  ```
  Control ImportantApplication
   {
       CONTAINER_NAME                    Top
       CONTAINER_TYPE                    BOX
       POSITION_HINTS                    6
    ...
  }
  ```

  个人的控制具有优先权，因此控制将被放置在位置 6。

指定控制所用的图标

控制定义的 ICON 字段定义了用于控制的图标图像。

ICON 字段的值可以是：

- 基本文件名。

  基本文件名是包含图标图像的文件的名称减去表示大小 (m 和 t) 和图像 类型 (bm 和 pm) 的文件名后缀。例如，如果文件名为 MyGame.l.pm 和 MyGame.m.pm， 则使用 MyGame。

  如果使用基本文件名，图标文件必须放置在图标查找路径的目录中：

  - 个人图标：*HomeDirectory* /.dt/icons
  - 系统范围图标：/etc/dt/appconfig/icons/*language*
- 图标文件的绝对路径，包括全文件名。

  只有在图标文件不在图标查找路径中的情况下，才应使用绝对路径。

  所需的大小图标取决于控制的位置：

  位置大小

  主面板–48 乘 48 像素 (*name*.l.pm 或 *name*.l.bm)

  子面板–24 乘 24 像素 (*name*.s.pm 或 *name*.s.bm)

  将图标文件放置在下面的位置之一：

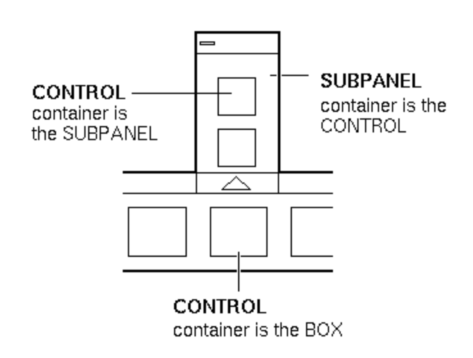
- 个人图标：*HomeDirectory* /.dt/icons
- 系统范围图标：/etc/dt/appconfig/icons/*language*

创建和修改子面板

用户可使用“前面板”弹出式菜单创建和修改子面板。

此节讨论如何提供系统范围的定制，这就需要修改“前面板”配置文件。

子面板被“附加”到“主面板”的控制中。



图形 15–3 子面板的容器是附加到容器的控制

附加在子面板定义中完成。CONTAINER_NAME 字段指定子面板附加到的控制：

```
CONTROL control_name
{
     ...
}

SUBPANEL subpanel_name
{
     CONTAINER_NAME                    control_name
  ...
}
```

▼ 创建系统范围的新子面板

1. **在想要附加子面板的主面板中定位控制的** *control_name*。

   如果控制是内置控制之一，它的定义在 /usr/dt/appconfig/types/*language* /dtwm.fp 中。

2. **创建新文件** /etc/dt/appconfig/types/*language* /*.fp。

3. **定义子面板：**

```
SUBPANEL subpanel_name
{
     CONTAINER_NAME          control_name
     TITLE          value
     KEYWORD          value
     . . .
}
```

4. **保存新的配置文件。**

5. **从"工作区"菜单中选择"重启工作区管理器"。**

定制内置子面板

可修改内置子面板的一般特性 (如标题之类) 和内容。

▼ 修改内置子面板的一般特性

1. **创建新的"前面板"配置文件：**

   - 系统范围：`/etc/dt/appconfig/types/`*language*`/`*name* `.fp`
   - 个人：*HomeDirectory* `/.dt/types/`*name*`.fp`.

2. **将整个缺省 `SUBPANEL` 定义从 `/usr/dt/appconfig/types/`*language*`/dtwm.fp` 复制到新的文件：**

   ```
   ^ISUBPANEL    subpanel_name
       {
         . . .
       }
   ```

3. **修改子面板定义。**

4. **保存新的配置文件。**

5. **从"工作区"菜单中选择"重启工作区管理器"。**

   例如，放置在文件 `/users/janice/.dt/types/PerApps.fp` 中的下面的定义， 会更改"个人应用程序"子面板的名称：

   ```
   SUBPANEL PersAppsSubpanel
    {
        CONTAINER_NAME                    TextEditor
        TITLE                               Janice's Applications
    }
   ```

▼ 向内置子面板添加系统范围的控制

1. **创建"前面板"配置文件 `/etc/dt/appconfig/types/`*language*/*name*`.fp`.**

2. **在文件中定义系统范围的控制。**

使用 CONTAINER_NAME 和 CONTAINER_TYPE 字段指定控制的容器：

```
CONTROL control_name
{
    CONTAINER_NAME                    subpanel_name
    CONTAINER_TYPE                    SUBPANEL
    ...
}
```

请参见 第 231 页"定义前面板控制"。

3. **保存配置文件。**

4. **从"工作区"菜单中选择"重启工作区管理器"。**

   例如，在新文件 /etc/dt/appconfig/types/*language* /DigitalClock.fp 中定义的下面的控制，向所有用户的"个人应用程序"子面板添加了 DigitalClock (在 Desktop _Tools 应用程序组中)。

```
CONTROL DigitalClockControl
 {
     TYPE                        icon
     CONTAINER_NAME             PerAppsSubpanel
     CONTAINER_TYPE             SUBPANEL
     ICON                        Dtdgclk
     PUSH_ACTION                DigitalClock
     PUSH_RECALL                True
 }
```

▼ 从内置子面板中删除控制

● **使用与删除"主面板"控制相同的过程。请参见第 225 页"删除控制"。**

▼ 删除安装图标控制

● **在子面板定义中添加下面的字段：**

```
CONTROL_INSTALL    False
```

▼ 更改子面板的自动关闭性能

用户选择控制时，除非用户从子面板的原始位置将其移动，否则子面板的缺省性能是关闭。

可配置"前面板"以保持子面板的打开状态，直到用户明确地关闭它们。

1. **在下面的位置创建新的"前面板"配置文件：**

   - 系统范围：/etc/dt/appconfig/types/*language* /*.fp

■ 个人：*HomeDirectory* /.dt/types/*.fp

2. **将缺省的** PANEL **定义从** /usr/dt/appconfig/types/*language*/dtwm.fp **复制到新的文件：**

```
PANEL FrontPanel
 {
   ...
 }
```

3. **向** PANEL **定义中添加下列字段：**

```
SUBPANEL_UNPOST    False
```

4. **保存新的配置文件。**

5. **从"工作区"菜单中选择"重启工作区管理器"。**

定义前面板控制

用户可通过在"安装图标"控制上拉放图标来创建个人控制。

虽然这样可提供简捷的可定制性，但所提供的功能却是"前面板"控制的功能的子集。例如，使用"安装图标"控制所创建的控制无法：

■ 提供动画

■ 显示客户机窗口

■ 在事件发生时 (例如，在收到新的邮件时) 更改外观

本节描述如何手动地创建"前面板"控制。

有关"前面板"控制的语法参考信息，请参见 dtfpfile(4X) 手册页。

前面板控制的定义

"前面板"控制定义的结构是：

```
CONTROL control_name
 {
   TYPE               control_type
   CONTAINER_NAME   value
   CONTAINER_TYPE   value
   other fields defining appearance and behavior}
```

控制类型

在控制定义中的 TYPE 字段指定了控制的基本性能。

控制 TYPE	控制性能
图标 (缺省的)	用户单击控制或在其上拉放文件时，它将运行指定的操作。
空格	用于调整控制间距的占位符。
忙	忙灯。调用操作时，该控制闪烁 (切换图像)
客户机	“前面板”中的客户机窗口。
时钟	时钟。
日期	显示当前日期。
文件	代表文件。选择控制将运行该文件的缺省操作。

▼ 创建新控制

本节描述定义控制的一般步骤，以及如何创建各种类型的控制。

1. **如果控制有一个 `PUSH_ACTION` 和/或 `DROP_ACTION`，则创建操作定义。这些是在用户单击控制或在其上拉放文件时运行的操作。**

2. **创建控制的图标图像文件。**

   有关图标大小、名称和位置的信息，请参见第 209 页“图标图像文件”。

3. **在下面的位置创建新的“前面板”配置文件：**

   - 系统范围：/etc/dt/appconfig/types/*language* /*.fp
   - 个人：*HomeDirectory* /.dt/types/*.fp

4. **在文件中添加控制定义。**

5. **保存文件。**

6. **从“工作区”菜单中选择“重启工作区管理器”。**

创建被单击时运行操作的控制

使用下面的字段定义控制的性能：

- TYPE：设置为图标
- PUSH_ACTION：指定要运行的操作名称

  例如，下面这一控制将放置到“个人应用程序”子面板，它运行用户已获得的游戏：

```
CONTROL Ball
 {
     TYPE                              icon
     CONTAINER_NAME                  PersAppsSubpanel
     CONTAINER_TYPE                  SUBPANEL
     ICON                              ball
     PUSH_ACTION                      RunBallGame
     HELP_STRING                      "选择此控制以播放 Ball。"
 }
```

下面的控制将位于 switch (开关) 的左上角。它启动名为 CutDisp 的操作。

```
CONTROL StartCutDisp
 {
   TYPE                              icon
   CONTAINER_NAME              Switch
   CONTAINER_TYPE              SWITCH
   POSITION_HINTS              first
   ICON                              cutdisp
   HELP_STRING                  "选择此控制以运行 cutdisp。"
      PUSH_ACTION                  CutDisp
 }
```

创建打开文件的控制

使用下面的字段定义控制的性能：

- TYPE：设置为文件
- FILE_NAME：指定要打开文件的路径
- PUSH_ACTION：设置为打开

  必须有为文件的数据类型而定义的"打开"操作。

  例如，下面的控制将位于"主面板"的最右边。它用数据文件 /users/ellen/PhoneList.txt 启动"文本编辑器"。*.txt 文件的"打开"操作是缺省的操作数据库的一部分。

```
CONTROL EditPhoneList
 {
     TYPE                              file
     FILE_NAME                    /users/ellen/PhoneList.txt
     CONTAINER_NAME                  Top
     CONTAINER_TYPE                  BOX
     POSITION_HINTS                  last
     ICON                              PhoneBook
     HELP_STRING                      "此控制显示 Ellen 的电话列表。"
     PUSH_ACTION                      Open
 }
```

创建作为拉放区的控制

当用户将一个文件拉放在控制上时，使用 DROP_ACTION 字段来指定运行的操作。操作必须能够接收文件变量。

常常，控制定义包括一个 PUSH_ACTION 字段和 DROP_ACTION 字段。对于按下和拉放操作可使用相同的操作。

例如，下面的位于"个人应用程序"子面板的控制运行 X 客户 xwud，它带有一个文件变量。

```
CONTROL Run_xwud
 {
     CONTAINER_NAME                PerAppsSubpanel
     CONTAINER_TYPE                SUBPANEL
     POSITION_HINTS                2
     ICON                            XwudImage
     PUSH_ACTION                    RunXwud
     DROP_ACTION                    RunXwud
 }
```

创建监视文件的控制

使用下面的字段定义控制的性能：

- TYPE：指定下面的值之一：

  图标—如要指定此控制的一个 PUSH_ACTION 和/或 DROP_ACTION，则使用此类型。

  文件—如果要使选中的控制在 "文件管理器"中双击文件的图标时表现得象文件，则使用此类型。

- ICON 和 ALTERNATE_ICON：描述用于表示被监视文件的未更改和已更改状态的图像。
- MONITOR_TYPE：描述引起图像更改的条件。使用下面的值之一：

  邮件—当向文件添加信息时，该控制将更改外观。

  文件—当指定文件变成非空时，该控制将更改。

- FILE_NAME：指定要监视的文件。

  例如，下面的控制寻找名为会议的文件的存在， 您希望使用匿名 ftp 将该文件传输到系统上。这个控制放置在"个人应用程序" 子面板的顶部。

  ```
  CONTROL MonitorCalendar
   {
       TYPE                              file
       CONTAINER_NAME                  PersonalApps
       CONTAINER_TYPE                  SUBPANEL
       POSITION_HINTS                  first
       FILE_NAME                      /users/ftp/meetings
  ```

```
        MONITOR_TYPE                file
        ICON                            meetingsno
        ALTERNATE_ICON                meetingsyes
    }
```

创建单一实例 (切换) 控制

单一实例控制检查由 PUSH_ACTION 启动的进程是否已在运行。如果该进程没有运行，则运行 PUSH_ACTION。如果该进程已在运行， 则将窗口移动到当前工作区窗口 stack (栈) 的顶部。

使用下面的字段定义控制的性能：

- PUSH_RECALL：设置为真。
- CLIENT_NAME：指定控制的客户机名称。

  CLIENT_NAME 的值必须与应用程序的顶级窗口上 WM_CLASS 特性中的第一个字符串 (*res_name*) 相匹配。有关详细信息，请参见 xprop(1) 手册页。
- PUSH_ACTION：描述用户单击控制时所运行的操作。

  例如，下面的控制运行应用程序的单一实例，其操作名为 MyEditor。

```
CONTROL MyEditor
  {
       TYPE        icon
   CONTAINER_NAME                Top
   CONTAINER_TYPE                BOX
   POSITION_HINTS                15
   PUSH_RECALL                    True
   CLIENT_NAME                    BestEditor
   PUSH_ACTION                    StartMyEditor
   ICON                            MyEd
  }
```

▼ 创建客户机窗口控制

客户机窗口控制是嵌入"前面板"中的应用程序窗口。例如，通过创建 xload 客户机窗口控制，可将系统负荷监测器放置在"前面板"中。

1. 定义控制。

使用下面的字段定义控制的性能：

- TYPE：设置为客户机。
- CLIENT_NAME：指定要启动的客户机。

  CLIENT_NAME 的值必须与应用程序顶层窗口上的 WM_CLASS 特性中的第一个字符串 (*res_name*) 相匹配。有关详细信息，请参见 xprop(1) 手册页。
- CLIENT_GEOMETRY：以像素为单位指定客户机"前面板"窗口所需的大小。

xwininfo(1) 手册页描述如何找出以像素为单位的窗口的大小。

2. **从“工作区”菜单中选择“重启工作区管理器”。**

3. **从终端仿真器的命令行启动客户机。**

   例如，下面的控制显示了 30 乘 20 像素的负荷监测器。

```
CONTROL LoadMeter
  {
   TYPE        client
   CONTAINER_NAME               Top
   CONTAINER_TYPE               BOX
   CLIENT_NAME                   xload
   CLIENT_GEOMETRY            30x20
  }
```

   如果未在会话之间保存和恢复客户机，您可能想配置控制，以便当用户单击它时启动客户机。例如，可以通过向定义添加下面的行，来配置“负荷监测器”控制，以便启动 `xload`：

```
 PUSH_ACTION                StartXload
```

   并创建操作：

```
ACTION StartXload
  {
   WINDOW_TYPE                 NO_STDIO
   EXEC_STRING                 /usr/contrib/bin/X11/xload
  }
```

▼ 启动动画控制

可附加动画序列，它将在用户选中控制或在其上拉放对象时使用。

为了具有动画序列，控制必须：

- 为类型图标
- 具有 PUSH_ACTION 或 DROP_ACTION

1. **使用 ANIMATION 组件指定动画序列：**

```
ANIMATION animation_name
       {
        ANIMATION    icon1                    [milisecond_delay]
        ANIMATION    icon1                    [milisecond_delay]
          ...
      }
```

   其中 *icon1*、*icon 2* 等等为图标的名称，而 *milisecond_delay* 是动画图标之间的以毫秒为单位的时间延迟。缺省的时间延迟为二百毫秒。

2. **将 PUSH_ANIMATION 和/或 DROP_ANIMATION 字段添加到控制定义中。该值是 ANIMATION 序列的名称。**

例如，下面的行激励了启动 BestEditor 应用程序的控制。图标间的时间延迟为三百毫秒。该实例假定已创建了图标文件框架 1、框架 2 等等。

```
CONTROL BestEditor
 {
     ...
     PUSH_ANIMATION BestEdAnimation
     ...
 }
 ANIMATION BestEdAnimation
 {
     frame1           300
     frame2
     ...
 }
```

提供前面板控制的针对项帮助

有两种提供控制帮助的方法：

- 在控制定义中提供帮助字符串。

  当用户调用控制的针对项帮助时，帮助字符串显示在帮助查看器中。帮助字符串不能包含格式 (如标题之类) 或链接。

  要提供帮助字符串，在控制定义中将其指定为：

  `HELP_STRING` *help_string*

- 在注册的帮助卷中指定帮助主题。

  帮助主题是使用帮助系统的全部能力而编辑的信息。编辑帮助主题需要使用桌面"帮助开发者工具包"。

  要提供帮助主题，请指定控制定义中的帮助卷和主题 ID：

  `HELP_VOLUME` *help_volume_name*
  `HELP_TOPIC` *topic_id*

定制工作区切换

有几种定制工作区切换的方法：

- 更改工作区的数量
- 更改切换的布局
- 更改切换中的控制

▼ 更改工作区的缺省数目

● **修改下面的“工作区管理器”资源：**

```
Dtwm*workspaceCount:                              n
```

有关详细信息，请参见第 246 页“更改系统范围基础上的工作区数目”.

▼ 更改切换的行数

● **修改 SWITCH 定义的 NUMBER_OF_ROWS 字段。**

例如，下面的定义定义了三行切换。

```
SWITCH Switch
  {
   CONTAINER_NAME                  box_name
   NUMBER_OF_ROWS                  3
  ...
  }
```

▼ 更改或添加工作区切换中的控制

1. **用控制定义创建“前面板”配置文件。**

   ■ 指定该控制在切换内部：

   ```
   CONTAINER_NAME   Switch
      CONTAINER_TYPE   SWITCH
   ```

   ■ 指定在切换中的位置：

   ```
           POSITION_HINTS     n
   ```

   其中 *n* 为整数。位置按从左到右、从上到下进行顺序编号。(对于缺省的两行切换，位置是从 1 到 4。)

2. **创建控制的图标。推荐的大小为 16 乘 16 像素。**

   例如，下面的控制在切换中放置了“终端”控制。

```
CONTROL SwitchTerminal
  {
       TYPE             icon
       CONTAINER_NAME                Switch
      CONTAINER_TYPE                SWITCH
      POSITION_HINTS                3
      ICON             Fpterm
      LABEL             Terminal
      PUSH_ACTION                    Dtterm
```

```
        HELP_TOPIC                          FPOnItemTerm
        HELP_VOLUME                          FPanel
    }
```

该控制使用内置图标及与“终端”控制使用相同的帮助主题，该终端控制存在于“个人应用程序”子面板中。

一般的前面板配置

“前面板”的 PANEL 语法使您可以：

- 更改“前面板”的位置
- 更改窗口布景
- 设置控制的一般外观和性能

  缺省的 PANEL 描述在 /usr/dt/appconfig/types/*language* /dtwm.fp 中。

  有关附加信息，请参见 dtfpfile(4X) 手册页。

一般步骤

1. **在 /etc/dt/appconfig/types/*language* 或 *HomeDirectory*/.dt/types 中创建新的“前面板”配置文件。**

2. **将缺省的 PANEL 描述从 /usr/dt/appconfig/types/*language* /dtwm.fp 复制到新文件中。**

3. **编辑 PANEL 描述。**

   新的 PANEL 描述具有高于缺省描述的优先级。

▼ 更改缺省的前面板位置

● **使用 PANEL 定义中的 PANEL_GEOMETRY 字段来指定位置。**

例如，下面的面板在右上角。

```
PANEL SpecialFrontPanel
  {
   PANEL_GEOMETRY       -1+1
   ...
  }
```

▼ 在主面板中标记控制

1. **向** PANEL **定义中添加下列行：**

   ```
   DISPLAY_CONTROL_LABELS    True
   ```

2. **对每个控制添加** LABEL **字段。**

   如果未指定 LABEL，则使用 *control_name*。

▼ 更改控制的单击性能

- **使用** PANEL **定义中的** CONTROL_BEHAVIOR **字段来指定用户如何运行控制的** PUSH _ACTION**。该字段值是：**

  单击—用户单击控制来运行 PUSH_ACTION

  双击—用户双击控制来运行 PUSH_ACTION

▼ 创建全新的前面板

要进行大量更改时，可能会首选创建新的"前面板"。

为了防止与内置的"前面板"组件冲突，一个全新的"前面板"应该对 PANEL 和其他容器使用新的名称。

1. **为新的"前面板"创建** PANEL **组件。给它一个唯一的名称：**

   ```
   PANEL front_panel_name
        {
             ...
         }
   ```

2. **用新的容器名称创建新的框和控制。**

   如果使用现有的组件，必须复制它们的定义并更改 CONTAINER_NAME 值。

3. **从"工作区"菜单中选择"重启工作区管理器"。**

创建具有三行的个人前面板实例

下面的实例更改了缺省的"前面板"，以便将它的控制组织到三行中。

1. **将** /usr/dt/appconfig/types/*language*/dtwm.fp **复制到** *HomeDirectory*/.dt/types/MyFrontPanel.fp **中。给文件写许可。**

   这是要编辑以便提供新的"前面板"的文件。

2. 更改“前面板”的名称：

```
PANEL NewFrontPanel
```

3. 更改名为 **Top** 的框的名称，并编辑它的容器名称：

```
BOX NewFrontPanelTop
  {
   CONTAINER_NAME                    NewFrontPanel
   POSITION_HINTS                    first
   ...
  }
```

4. 为中间和底部的行添加框定义：

```
BOX NewFrontPanelMiddle
  {
   CONTAINER_NAME            NewFrontPanel
   POSITION_HINTS            second
  }


BOX NewFrontPanelBottom
  {
   CONTAINER_NAME        NewFrontPanel
          POSITION_HINT        second
  }
```

5. 将下列控制的 `CONTAINER_NAME` 改为 `NewFrontPanelTop`：

- 时钟
- 日期
- 起始
- 文本编辑器
- 邮件

6. 将下列控制的 `CONTAINER_NAME` 改为 `NewFrontPanelBottom`：

- 打印机
- 式样
- 应用程序
- 帮助
- 垃圾

7. 将切换的 `CONTAINER_NAME` 改为 `NewFrontPanelMiddle`。

8. 从“工作区”菜单中选择“重启工作区管理器”。

第 16 章

定制工作区管理器

本章描述如何定制桌面“工作区管理器”。

“工作区管理器”是桌面提供的窗口管理器。 类似于其他的窗口管理器，它控制：

- 窗框组件的外观
- 窗口的性能，包括它们的堆叠次序和聚焦性能
- 键绑定和按钮绑定
- 最小化窗口的外观
- “工作区”和“窗口”菜单

另外，“工作区管理器”控制这些桌面组件：

- **工作区**。 “工作区管理器”控制工作区的数目，并跟踪每个工作区中所打开的窗口。
- **工作区背景**。用户使用“式样管理器”更改背景。但是，背景管理是“工作区管理器”的一个功能。
- **前面板**。尽管“前面板”使用自身的配置文件，但是，它由“工作区管理器”创建和管理。

其中许多可以用“式样管理器”更改。“式样管理器”能够使常用更改快速而不费力。其他资源必须手动设置。

“工作区管理器”是 `dtwm`。它基于“Motif 窗口管理器”。

- 有关“工作区管理器”的参考信息，请参见 dtwm(1) 和 dtwmrc(4) 手册页。
- 有关设置“工作区管理器”资源的信息，请参见第 257 页“设置应用程序资源”。
- 有关“前面板”配置文件的信息，请参见第 15 章。

有关设置资源的附加信息，请参见第 257 页“设置应用程序资源”。

工作区管理器配置文件

“工作区管理器”从配置文件得到有关窗口菜单、工作区菜单、按钮绑定以及键绑定的信息。

它使用下面的文件之一：

- 个人文件： *HomeDirectory* /.dt/dtwmrc
- 系统定制文件：/etc/dt/config/*language* /sys.dtwmrc
- 内置文件： /usr/dt/config/*language* /sys.dtwmrc

“工作区管理器”以上述次序搜索配置文件， 并使用最先找到的那个。

对于使用不止一个会话语言的用户，可以创建个人的、与语言相关的配置文件 *HomeDirectory* /.dt/*language* /dtwrmc，其优先级高于 *HomeDirectory* /.dt/dtwmrc。

▼ 创建或修改个人配置文件

个人的工作区管理器配置文件是 *HomeDirectory* /.dt/dtwmrc。 若该文件存在，则它就是所使用的文件。

1. **双击 Desktop_Tools 应用程序组中的 Edit Dtwmrc 。**

   如果已有个人的 dtwmrc 文件，它就被装入编辑器。如果没有，那么 sys.dtwmrc 被复制到 *HomeDirectory*/.dt/dtwmrc 中，然后装入编辑器。

2. **编辑文件。**

3. **退出编辑器。**

   不管文件最初来源如何，它都被保存为个人 dtwmrc。

▼ 创建系统范围的配置文件

系统范围的“工作区管理器”配置文件是 /etc/dt/config/*language* /sys.dtwmrc。

- **将** /usr/dt/config/*language* /sys.dtwmrc **复制到** /etc/dt/config/*language* /sys.dtwmrc。

注意：如果 *HomeDirectory*/.dt/dtwmrc 存在，则不使用此文件。

▼ 包括 (源) 其他文件

- **使用语法：**

```
include
 {
 path
path
   ...
 }
```

例如，下面的行源在文件 /users/ellen/mymenu 中：

```
include
    {
      /users/ellen/mymenu
  }
```

对于没有复制整个配置文件提供的附加功能而言，包括语句很有用。例如，用户可能想没有管理整个配置文件而创建新的键绑定。用户可以创建含有下面的内容的文件 *HomeDirectory*/.dt/dtwmrc：

```
include
 {
  /etc/dt/config/C/sys.dtwmrc
 }
Keys DtKeyBindings
 {
   Alt<Key>F5 root  f.menu  Applications

 }

Menu Applications
  {
   "GraphicsApp" f.exec "/usr/bin/GraphicsApp/GApp"
   ...
  }
```

▼ 重新启动工作区管理器

为了使对配置文件所作的更改生效，必须重新启动"工作区管理器"。

- **从"工作区"菜单选择"重新启动工作区管理器" (当指针落在背景上时，按鼠标按钮 3)。**

定制工作区

用户可以用桌面的界面完成大多数工作区定制，比如更改工作区名称和工作区的数目等。但是，“工作区管理器”为设置系统范围的缺省值提供了资源。

▼ 更改系统范围基础上的工作区数目

缺省的桌面配置提供四个工作区。用户可使用与工作区切换相关联的弹出式菜单来添加和删除工作区。

/usr/dt/app-defaults/C/Dtwm 文件将 workspaceCount 资源设置成下面的缺省数目的工作区：

```
Dtwm*0*workspaceCount: 4
 Dtwm*workspaceCount:  1
```

在屏幕 0 上指定多个工作区。在任何其他屏幕上指定单个的工作区。

可创建 (或修改现有的) /etc/dt/config/C/sys.resources 文件，以便为工作站上所有新用户更改缺省的工作区数目。

- **使用** 0*workspaceCount **资源在主屏幕上设置系统范围的缺省值**：

```
Dtwm*0*workspaceCount: 数目
```

例如，下面的资源在主屏幕上设置系统范围的工作区数目为 6 ：

```
Dtwm*0*workspaceCount: 6
```

有关设置“工作区管理器”资源的信息，请参见第 257 页“设置应用程序资源” 。

例如，下面的资源设置工作区数目为 6 ：

```
Dtwm*workspaceCount: 6
```

▼ 提供系统范围的工作区名称

在内部，工作区按编号惯例 ws*n* 编号，其中 *n* 是 0、1、2 等等。例如，缺省的四个工作区在内部被编号为 ws0 到 ws3。

- **使用**标题**资源以更改指定工作区的名称**：

```
Dtwm*wsn: 名称
```

有关设置“工作区管理器”资源的信息，请参见第 257 页“设置应用程序资源”.

例如，下面的资源将缺省的四个工作区设置为指定的名称：

```
Dtwm*ws0*title:    Anna
Dtwm*ws1*title:    Don
Dtwm*ws2*title:    Julia
Dtwm*ws3*title:    Patti
```

▼ 创建附加的背景

1. **创建背景图像。它们可以是位图或像素图文件。**
2. **将背景放进下面的目录之一。(可能需要创建此目录。)**
   - 系统范围背景：/etc/dt/backdrops
   - 个人背景：*HomeDirectory*/.dt/backdrops
3. **从"工作区"菜单中选择"重新启动工作区管理器"。**

   将系统范围和个人背景添加到 /usr/dt/backdrops 的内置背景中。

   可以创建与其同名的个人或系统范围背景，来替换现有的内置背景。

▼ 用图形图像替换背景

背景被分层在显示的根窗口之上。"式样管理器背景"对话框提供"无背景"设置，在此设置下背景是透明的。

在所有的工作区背景后面只有一个根窗口。因此，放置在根窗口中的图形图像始终跨越在所有的工作区。可以指定哪些工作区用背景覆盖根窗口。然而，当"无背景"有效时，对每个工作区可见的图像是相同的。

1. **创建图形图像。**

   图像必须是有工具能够在根窗口中将其显示出来的格式。例如，如果要使用 xsetroot，必须创建一个位图文件。

2. **如果它还不存在，就要创建一个可执行文件** *HomeDirectory* **/.dt/sessions/sessionetc。**

   sessionetc 文件在用户每次登录时运行。

3. **在 sessionetc 文件中放置用于显示图像的命令。**

   例如，下面的命令用指定的位图平铺显示根窗口：

   ```
   xsetroot -bitmap /users/ellen/.dt/icons/root.bm
   ```

工作区管理器菜单

“工作区管理器”有三个缺省菜单：

工作区菜单—又称作根菜单。当指针位于背景上时，用户按鼠标按钮 3，即显示该菜单。该菜单通过按钮绑定与鼠标按钮相关联。

窗口菜单—用户在指针位于“窗口”菜单按钮 (窗框的左上角) 上时按鼠标按钮 1 或 3，即显示该菜单。该菜单通过 `windowMenu` 资源与按钮相关联。

“前面板”菜单—用户在指针位于“前面板”的“窗口”菜单按钮上时按鼠标按钮 1 或 3，即显示该菜单。

工作区管理器菜单语法

“工作区管理器”菜单具有下面的语法：

```
Menu MenuName
 {
      selection1  [mnemonic]  [accelerator] function [argument]
      selection2  [mnemonic]  [accelerator] function [argument]
      ...
 }
```

其中：

selection—出现在菜单中的文本或位图。如果文本包含空格，就用引号将文本括起。对于位图，使用语法 @/*path*。

mnemonic—是显示菜单时用作键盘简捷方式的单个字符。在 form _*character* 中指定。

accelerator—无论菜单是否显示，键盘简捷方式都是活动的。加速器具有语法 *modifier* <Key> *Keyname*，其中 modifier 是 `Ctrl`、 `Shift`、`Alt` (扩展字符) 或 `Lock`。 有关所有可能键名称的列表，请参见 “X11 include”目录中的 `keysymdef.h` 文件。

function—作出该选择后要执行的功能。有关功能的列表，请参见 `dtwmrc`(4) 手册页。

argument—函数变量。有关详细细节，请参见 dtwmrc(4) 手册页。

例如，下面的标记为“恢复”的菜单项使窗口 normalize (规范化)。在显示菜单时，请键入“R”也将恢复窗口。按“扩展”字符 F5 键也将恢复窗口。

```
Restore    _R    Alt<Key> F5    f.normalize
```

注意：有关“工作区管理器”菜单语法的完整信息，请参见 `dtwmrc`(4) 手册页。

▼ 在工作区菜单添加新的菜单项

1. **单击"前面板"的"工具"子面板中的"向菜单添加项"控制。**

   显示"向工作区菜单添加项"对话框。



2. **键入完整路径，或者单击"浏览"并选择要运行新菜单项的文件的路径。**

   所指向的文件在正常情况下应该可以为该主机所用。它应该是可执行文件或是已注册到主机的数据类型文件 (例如，"音频"或"手册页")。

   包含文件名的完整路径显示在文本框中。

3. **单击"添加"，向菜单中添加文件。**

   文件被添加到"工作区菜单"的第一个位置。它的缺省图标 (如有) 和文件名将用作菜单项的图标和文本。

注意：要更改"工作区菜单"中的菜单项位置，请参见第 249 页"修改工作区菜单"。

▼ 修改工作区菜单

1. **单击"前面板"的"工具"子面板上的"定制工作区菜单"控制。**

   所显示"文件管理器"显示出"工作区菜单"的配置文件夹。注意，该文件夹中的文件代表每个菜单项，子文件夹代表每个子菜单。 通过重新安排"工作区菜单"文件夹的内容，可重新安排"工作区菜单"。



2. **将任何想要移动的菜单项拖放到该子文件夹下的新位置。**

   有关在“树视图”模式下使用“文件管理器”的帮助，请参见《*Solaris* **公共桌面环境：用户指南**》。

3. **把不需要的菜单项文件拖到“前面板”的“垃圾篓”中，将其删除。**

4. **通过编辑其文件或文件夹名称，来重命名需要重命名的任何菜单项。**

5. **对作出的更改感到满意后，从"文件管理器"的"文件"菜单中选择"更新工作区菜单"，并退出"文件管理器"。**

   "工作区菜单"现在反映出在"工作区菜单"文件夹中所作的更改。

▼ 创建新的工作区 (根) 菜单

1. **打开相应的文件，以便进行编辑：**

   - 个人：*HomeDirectory* /.dt/dtwmrc
   - 系统范围：/etc/dt/config/*language* /sys.dtwmrc

     有关创建这些文件的信息，请参见第 244 页"工作区管理器配置文件"。

2. **创建新菜单：**

```
Menu menu_name
{
          ...
 }
```

   请参见 第 248 页"工作区管理器菜单语法"。

3. **创建或编辑按钮绑定来显示新菜单。**

   如果该菜单替换了现有的菜单，则编辑显示"工作区"菜单的键绑定。

```
<Btn3Down> root  f.menu  menu_name
```

   如果该菜单是附加菜单，则创建一个新的鼠标按钮绑定。例如，在背景上同时按 Shift 键和鼠标按钮 3 ，下面的按钮绑定就显示该菜单：

```
Shift<Btn3Down> root  f.menu  menu_name
```

4. **从"工作区"菜单中选择"重新启动工作区管理器"。**

▼ 创建新的窗口菜单

注意："窗口"菜单内置在"工作区管理器"中，通常不被定制。为保持窗口性能在应用程序间的一致性，应该避免对"窗口"菜单进行大量的修改。

---

1. **打开相应的文件，以便进行编辑：**

   - 个人： *HomeDirectory* /.dt/dtwmrc
   - 系统范围：/etc/dt/config/*language* /sys.dtwmrc

     有关创建这些文件的信息，请参见第 244 页"工作区管理器配置文件"。

2. **创建新的菜单：**

```
Menu menu_name
{
        ...
}
```

3. **使用 windowMenu 资源指定新的菜单：**

```
Dtwm*windowMenu: menu_name
```

4. **从"工作区"菜单中选择"重新启动工作区管理器"。**

定制按钮绑定

按钮绑定及可能的键盘修改键与窗口管理器功能相关联。按钮绑定适用于所有的工作区。

桌面缺省按钮绑定定义在"工作区管理器"配置文件中的名为 DtButtonBindings 按钮绑定集之中：

```
按钮 DtButtonBindings
  {
    ...
  }
```

按钮绑定语法

按钮绑定的语法是：

```
Buttons ButtonBindingSetName
{
     [modifier] < button_nameMouse_action>   context function   [argument]
     [modifier] < button_nameMouse_action>   context function   [argument]
```

其中：

button_name —Btn1—鼠标左键 Btn2—中间键 (三键鼠标) 或两键 (双键鼠标) Btn3—右键 Btn4—在三键鼠标上同时按住键 1 和键 2 Btn5—在三键鼠标上同时按住键 2 和键 3

modifier —Ctrl、Shift、Alt、Lock

mouse_action —Down—按住鼠标；键 Up—释放鼠标按钮；Click—按下然后释放鼠标键；Click2—双击鼠标键； Drag—当按住鼠标键时拖动鼠标

context—表示绑定生效时指针必须在的位置。必要时，使用" | "字符将多个内容分开。

root—工作区窗口；window—客户窗口或窗框；frame—窗框架，不包括内容；icon—图标；title—标题栏；app—客户窗口 (不包括框架)

function—窗口管理器功能之一。有关有效功能的列表，请参考 dtwmrc(4) 手册页。

argument—任何需要的窗口管理器的功能变量。有关详细信息，请参考 dtwmrc(4) 手册页。

例如，当指针位于工作区窗口 (而不在客户窗口) 中，按鼠标键 3 时，下面的行显示 DtRootMenu 中描述的菜单。

```
<Btn3Down>       root      f.menu      DtRootMenu
```

注意：有关按钮绑定语法的完整信息，请参见 dtwmrc(4) 手册页。

▼ 添加按钮绑定

1. **打开相应的文件，以便进行编辑：**
   - 个人： *HomeDirectory* /.dt/dtwmrc
   - 系统范围：/etc/dt/config/*language* /sys.dtwmrc

   有关创建这些文件的信息，请参见 第 244 页"工作区管理器配置文件"。

2. **将按钮绑定添加到 DtButtonBindings 定义中。**

   不要将相同按钮绑定到单击和按下操作的不同的功能。也不要将多个功能绑定到相同的按钮和上下文。

3. **从"工作区"菜单中选择"重新启动工作区管理器"。**

▼ 创建新的按钮绑定集

1. **打开相应的文件，以便进行编辑：**
   - 个人： *HomeDirectory* /.dt/dtwmrc
   - 系统范围：/etc/dt/config/*language* /sys.dtwmrc

   有关创建这些文件的信息，请参见第 244 页"工作区管理器配置文件"。

2. **创建新的按钮绑定集。请参见 第 252 页"按钮绑定语法"。**

3. **将 buttonBindings 资源设置为新的名称：**

   Dtwm*buttonBindings: *ButtonBindingsSetName*

4. **从"工作区"菜单中选择"重新启动工作区管理器"。**

注意：新的按钮绑定将替换现有的按钮绑定。从 DtButtonBindings 复制要保留的按钮绑定。

定制键绑定

键盘绑定，也称作**键绑定**，将键的组合与“工作区管理器”功能相关联。 键绑定适用于所有工作区。

注意：使用公共键组合作为键盘绑定时要小心。例如，Shift-A 一般是在当前窗口中输入字母“A”。如果将 Shift-A 绑定到某功能，将失去它的正常用法。

缺省的桌面键绑定

桌面缺省的键绑定定义在“工作区管理器”配置文件中，在名为 DtKeyBindings 键绑定集中：

```
Keys DtKeyBindings
  {
    ...
  }
```

键绑定语法

键绑定的语法是：

```
Keys  KeyBindingSetName
 {
      [Modifiers] <Key>key_name  context   function    [argument]
      [Modifiers] <Key>key_name  context   function    [argument]
      ...
 }
```

其中：

Modifiers—Ctrl、Shift、 Alt 和 Lock。允许多个修饰符；用空格把它们分开。

key_name—功能所映射的键。对于字母键或数字键，*key_name* 名称通常打印在键上。例如，“a”键的名称为“a”，而“2”键的名称为“2”。“Tab”键名为“Tab”。“F3”键名为“F3”。

至于其他键，其名称被拼写出来—例如，plus 为“+”键。位于与系统相关的目录中的文件 keysymdef.h，包含了有关键名的附加信息。

context—该元素必须具有使该操作生效的键盘焦点。如果绑定适用于不止一个上下文，则这些可并置在一起。多个上下文用“|”字符分开。

`root`—工作区背景； `window`—客户窗口； `icon`—图标

function—窗口管理器功能。有关有效功能的列表，请参考 `dtwmrc`(4) 手册页。

argument—任何需要的窗口管理器的功能变量。有关详细信息，请参考 `dtwmrc`(4) 手册页。

例如，下面的键绑定使用户通过按 Alt+F6 ，将键盘焦点切换到应用程序中下一个瞬态窗口。

```
Alt<Key>F6      window      f.next_key      transient
```

注意：有关键绑定的完整信息，请参见 dtwmrc(4) 手册页。

▼ 创建定制键绑定集

1. **打开相应的文件，以便进行编辑：**
   - 个人： *HomeDirectory* `/.dt/dtwmrc`
   - 系统范围：`/etc/dt/config/`*language* `/sys.dtwmrc`

   有关创建这些文件的信息，请参见第 244 页“工作区管理器配置文件”。

2. **用专用的 *KeyBindingSetName* 创建新的键绑定集。使用桌面缺省的键绑定集 `DtKeyBindings` 作为指南。**

3. **将 `keyBindings` 资源设置为新的集名称：**

   `Dtwm*keyBindings:` *KeyBindingSetName*

4. **从“工作区”菜单中选择“重新启动工作区管理器”。**

注意：新的键绑定将替换现有的键绑定。将任何要保留的键绑定从 `DtKeyBindings` 复制到新集中。

在缺省和定制的性能之间切换

在 Motif 缺省和 CDE 桌面窗口性能之间切换：

1. **按 Alt+Shift+Ctrl+!**

2. **在对话框中单击"确定"。**

   切换到缺省性能的操作会去除"前面板"和任何定制键和按钮绑定。

第 17 章

管理应用程序资源、字体和颜色

通过使用"式样管理器"或者通过定制附加字体和颜色资源，可为显示器选择广泛的颜色和字体范围。本章讨论如何定制字体和颜色资源。

本章还描述如何指定 DtEditor 小部件应用程序，例如桌面文本编辑器 (dtpad) 和邮件程序 (dtmail) 的式样转换以及与这些转换冲突的 DtEditor 小部件应用程序菜单加速器的替换物。

设置应用程序资源

资源被应用程序用以设置外观和性能的某种特征。例如，"式样管理器"(dtstyle)提供这样的资源，它使您可以指定系统在何处寻找包含颜色调色板信息的文件：

```
dtstyle*paletteDirectories: /usr/dt/palettes/C  HomeDirectory/.dt/palettes
```

桌面应用程序的 App-default 文件位于 /usr/dt/app-defaults/*language* 目录。

▼ 设置系统范围的资源

● **向文件** /etc/dt/config/*language* /sys.resources **添加资源。 (可能必需创建文件。)**

例如，如果在 /etc/dt/config/C/sys.resources 中指定：

```
AnApplication*resource: value
```

则在下次登录时，资源 AnApplication*resource 将被设置在每个用户的 RESOURCE_MANAGER 特性中。

注意：有关“式样管理器”资源的详细信息，请参见 dtstyle 手册页。有关“邮件程序”资源的详细信息，请参见 dtmail 手册页。

▼ 设置个人资源

1. **将资源文件添加到文件** *HomeDirectory*/.Xdefaults。

2. **双击 Desktop_Tools 应用程序组中的“重新装入资源”。**

桌面如何装入资源

资源在“会话管理器”启动会话时装入。有关“会话管理器”如何将资源装入 RESOURCE _MANAGER 的信息，请参见，第 42 页“装入会话资源”。

进程管理器资源

下面的进程管理器资源可用：

- sampleNowTR
- postPopupMenuTR
- selectNextProcessTR
- selectPrevProcessTR
- selectFirstProcessTR
- selectLastProcessTR
- killSelectedProcessTR

定义 UNIX 绑定

缺省时 UNIX 绑定不可用。

▼ 指定 EMACS 式样转换

下面的过程指定了：

- `DtEditor` 小部件应用程序，例如桌面文本编辑器 (`dtpad`) 和邮件程序 (`dtmail`) 的 EMACS 式样转换
- 与这些转换冲突的 `DtEditor` 小部件应用程序菜单加速器的替换物。

1. **在文件** *HomeDirectory* `/.Xdefaults` **中添加下行：**

   `#include"/usr/dt/app-defaults/language/UNIXbindings"`

   其中 *language* 是 `LANG` 环境变量的值。

2. **重新启动会话。**

▼ 修改 EMACS-Style 转换

1. **将文件** `/usr/dt/app-defaults/`*language* `/UNIXbindings` **的内容插入** *HomeDirectory*`/.Xdefaults`**。**
2. **在** `.Xdefaults` **文件中编辑绑定。**
3. **完成后重新启动会话。**

由 UNIXbindings 文件提供的 UNIXbindings 绑定

`/usr/dt/app-defaults/`*language* `/UNIXbindings` 文件提供的绑定如下表所述。

注意：当 UNIX 绑定启用时，"Delete"键删除前一个字符，而 "Shift-Delete"键删除下一个字符。

表 17–1 列出和 UNIX 绑定冲突的菜单加速器和加速器文本的 `dtpad` 覆盖。

表 17–1 `dtpad` 覆盖

菜单加速器和加速器文本	覆盖
`Dtpad*fileMenu.print.acceleratorText:`	
`Dtpad*fileMenu.print.accelerator:`	
`Dtpad*editMenu.undo.acceleratorText:`	Ctrl+_
`Dtpad*editMenu.undo.accelerator:`	Ctrl<Key>_
`Dtpad*editMenu.paste.acceleratorText:`	Shift+Insert
`Dtpad*editMenu.paste.accelerator:`	Shift<Key>osfInsert

表 17–1 dtpad 覆盖 （续下）

菜单加速器和加速器文本	覆盖
Dtpad*editMenu.findChange.acceleratorText:	Ctrl+S
Dtpad*editMenu.findChange.accelerator:	Ctrl<Key>s

表 17–2 列出和 UNIX 绑定冲突的菜单加速器和加速器文本的 dtmail 创作窗口的覆盖。

表 17–2 dtmail 创作窗口覆盖

菜单加速器和加速器文本	覆盖
Dtmail*ComposeDialog*menubar*Edit.Undo.acceleratorText:	Ctrl+_
Dtmail*ComposeDialog*menubar*Edit.Undo.accelerator:	Ctrl<Key>_
Dtmail*ComposeDialog*menubar*Edit.Paste.acceleratorText:	Shift+Insert
Dtmail*ComposeDialog*menubar*Edit.Paste.accelerator:	Shift<Key>osfInsert
Dtmail*ComposeDialog*menubar*Edit.Find/Change.acceleratorText:	Ctrl+S
Dtmail*ComposeDialog*menubar*Edit.Find/Change.accelerator:	Ctrl<Key>s

下面的转换提供 (GNU 式样) EMACS 控制和元键绑定以及一些附加绑定。在适当的时候，它们也允许“Shift”键和正常绑定进行组合使用，以便反转操作的方向。例如，“Ctrl+Shift+F” 键将使光标后移一个字符，而正常情况下，“Ctrl+F” 键 使光标前移一个字符。

附加绑定是：

Ctrl+comma—后移一个单词

Ctrl+Shift+comma—前移一个单词

Ctrl+period—前移一个单词

Ctrl+Shift+period—后移一个单词

Ctrl+Return—文件结尾

Ctrl+Shift+Return—文件开头

GNU EMACS 将 delete-previous-character()，而不是 delete-next-character() 绑定到“Delete”键。正常情况下，“Meta+F”是文件菜单的助记符，所以将忽略 forward-word() 的绑定。使用 forward-word 的其它一种绑定 (例如，Ctrl+period)。

表 17–3 列出 DtEditor.text 转换

表 17–3 `DtEditor.text` 转换

修饰符键	键	操作例程
c ~s	<Key>a:	行开头()\n\
c s	<Key>a:	行结尾()\n\
c ~s	<Key>b:	向后一个字符()\n\
c s	<Key>b:	向前一个字符()\n\
c ~s	<Key>b:	向后一个字符()\n\
c s	<Key>b:	向后一词()\n\
m ~s	<Key>b:	向后一词()\n\
m s	<Key>b:	向前一词()\n\
c ~s	<Key>d:	删除下一个字符()\n\
c s	<Key>d:	删除前一字符()\n\
m ~s	<Key>d:	kill (删除) 下一个词()\n\
m s	<Key>d:	kill (删除) 前一个词()\n\
c ~s	<Key>e:	行结尾()\n\
c s	<Key>e:	行开头()\n\
c ~s	<Key>f:	向前一个字符()\n\
c s	<Key>f:	向后一个字符()\n\
m ~s	<Key>f:	向前一词()\n\
m s	<Key>f:	向后一词()\n\
c	<Key>j:	新行并缩进()\n\
c ~s	<Key>k:	kill (删) 到行结尾()\n\
c s	<Key>k:	kill (删) 到行开头()\n\
c	<Key>l:	重画显示()\n\
c	<Key>m:	新行()\n\
c s	<Key>n:	进程继续()\n\
c ~s	<Key>n:	进程关闭()\n\
c	<Key>o:	新行并备份()\n\
c ~s	<Key>p:	进程继续()\n\
c s	<Key>p:	进程关闭()\n\

表 17–3 DtEditor.text 转换 （续下）

修饰符键	键	操作例程
c ~s	<Key>u:	kill (删) 到行开头()\n\
c s	<Key>u:	kill (删) 到行结尾()\n\
c ~s	<Key>v:	下一页()\n\
c s	<Key>v:	前一页()\n\
m ~s	<Key>v:	前一页()\n\
m s	<Key>v:	下一页()\n\
c	<Key>w:	kill (删除) 选择部分()\n\
c ~s	<Key>y:	unkill (不删除) ()\n\
m	<Key>]:	向前一段()\n\
m	<Key>[:	向后一段()\n\
c ~s	<Key>comma:	向后一词()\n\
c s	<Key>comma:	向前一词()\n\
m	<Key>\\<:	文件开头()\n\
c ~s	<Key>period:	向前一词()\n\
c s	<Key>period:	向后一词()\n\
m	<Key>\\>:	文件结尾()\n\
c ~s	<Key>Return:	文件结尾()\n\
c s	<Key>Return:	文件开头()\n\
~c ~s ~m ~a	<Key>osfDelete:	删除前一字符()\n\
~c s ~m ~a	<Key>osfDelete:	删除下一字符()

管理字体

使用“式样管理器字体”对话框，可以选择为所有应用程序所要的字体组和字体大小。也可以在命令行中指定字体，或使用资源来：

- 设置单个应用程序的字体资源
- 指定不同字体供“字体”对话框使用

字体是一种文本字符被打印和显示的类型式样。桌面包含各种不同式样和大小的字体。

位图字体是由点的矩阵组成。(缺省情况下，“式样管理器”只配置位图字体。) 字体完全包含在一个文件中。要有一整套不同的大小、倾斜度和磅值，需要很多文件。

字体被指定为资源的值和命令的参数。“X 逻辑字体描述” (XLFD) 名称是请求所需字体的方法。系统查找到与它所给的描述最匹配的字体。

设置桌面字体资源

“式样管理器字体”对话框使您可为类似文本输入和标签这样的事物选择字体 (多达七种大小)。也可使您添加或删除字体组。

由字体对话框设置的资源

选中字体后，下面的资源写到 `RESOURCE_MANAGER` 特性中：

- 系统字体 用于系统区域，象菜单条、菜单窗格、按钮、切换按钮以及标签。下面的资源由系统字体设置：

  `*FontList`—显示在桌面客户机和用 OSF/Motif 工具包创建的其他客户机的系统区域中。

- 用户字体 用于窗口的文本输入。下面的资源由用户字体设置：

  `*Font`—支持 X 应用程序的较早版本

  `*FontSet`—主设置

  `*XmText*FontList`—显示在文本输入框中

  `*XmTextField*FontList`—显示在文本输入框中

字体对话框所使用的资源

用于字体对话框中每个选择的字体在 `/usr/dt/app-defaults/Dtstyle` 资源文件中指定。最多可指定七种大小。

`NumFonts`—字体对话框中字体大小的数字。

`SystemFont[1-7]`—最多七种资源，将特定字体指定给某一“字体”对话框选择，以作为系统字体

`UserFont[1-7]`—最多七种资源，将特定字体指定给某一“字体”对话框选择，以作为用户字体

注意：已选择了这些资源的缺省字体，以便满足在各种显示器上的可读性。如果要在应用程序中使用特定字体，请使用应用程序字体资源设置字体，而不要更改这些桌面字体。

有关应用程序字体的详细信息，请参见 DtStdAppFontNames(5) 和 DtStdInterfaceFontNames(5) 手册页。

▼ 列出可用的字体

1. **键入下面的命令：**

   `xlsfonts [-`*options*`] [-fn `*pattern*`]`

   显示了系统中可用的 XLFD 名称和字体别名的列表。位图字体在全部 14 个 XLFD 字段中的显示值。可缩放字样在 *PixelSize*, *PointSize*, *ResolutionX*和 *ResolutionY* 位置显示 0。

2. **要检查特定字体，请使用 `xlsfonts` 的模式匹配能力。在不打算匹配的模式部分使用通配符来代替。**

3. **如果 `xlsfonts` 未显示任何以 `dt` 开头的字体名，则字体路径未包含桌面字体。键入下面的命令，将桌面字体包含到可用字体中：**

   `xset +fp` **目录名**

   其中**目录名**是包含桌面字体的目录。会话启动所设置的缺省位置为 `/usr/dt/config/xfonts/`*language*.

   有关附加信息：

   - xset 和 xlsfonts 手册页列出可用的选项。
   - **使用** *X Window* **系统**解释字体别名和 `xset` 客户。

▼ 在命令行上指定字体

- **使用 `-xrm` 命令行选项来指定特定客户的字体资源。例如：**

application name `-xrm "*bitstream-charter-medium-r-normal-8-88-75-75-p-45-iso8859-1"`

X 逻辑字体描述 (XLFD)

通过列出 14 个不同的字符来指定字体，每个字符间用破折号 (-) 隔开。这称作 X 逻辑字体描述 (XLFD)。在某些情况下，列表中的特性可由一个 *通配符代替， 并且，特性中的一个字符可由一个 ? 通配符代替。表 17–4 列出字体特性字符串说明。

特性字符串说明的形式是：

"*-Foundry-FamilyName-WeightName- Slant-SetwidthName-AddStyleName- PixelSize-PointSize-ResolutionX-ResolutionY-Spacing- AverageWidth-CharSetRegistry- CharSetCoding*
"

表 17–4 字体特性字符串说明

特性字符串	定义
Foundry	标识字体设计者的字符串
FamilyName	标识字体的注册商标名称的字符串
WeightName	给出字体的相对磅值 (例如粗体) 的字符串
Slant	描述倾斜方向的代码： R (Roman–无倾斜) I (斜体–向右倾斜) O (倾斜–向右倾斜) RI (反转斜体–向左倾斜) RO (反转倾斜–向左倾斜)
SetwidthName	描述宽度 (例如压缩或扩展) 的字符串
AddStyleName	提供唯一标识字体所需要的任何附加信息的字符串
PixelSize	以像素为单位，给出 em-square 大小的整数
PointSize	以小数点为单位，给出 em-square 大小的整数
ResolutionX	以像素为单位，给出水平分辨率的整数
ResolutionY	以像素为单位，给出垂直分辨率的整数
Spacing	指定单元间的间隔的代码： M (等间隔--固定间距) P (比例间隔--变化间距) C (字符单元)
AverageWidth	以 1/10 个像素为单位，给出平均宽度的整数
CharSetRegistry	标识已注册了字体编码的注册授权的字符串
CharSetEncoding	在指定注册中标识字符集的字符串

实例

下面的 XLFD 名称描述了名为 charter 的字体，它是由支持 ISO8859-1 标准编码的 Bitstream 生成的：

```
-bitstream-charter-medium-r-normal--8-80-75-75-p-45-iso8859-1
```

它具有中等磅值，没有特殊倾斜，并且为正常宽度。字体是成比例的，具有 8 个像素或 8个点的 em-square 框。水平和垂直分辨率都是 75 个像素。字符的平均宽度为 45 个 1/10 像素或 4.5 个像素。

此字符串的一部分可由通配符代替。系统使用找到的第一个与您指定的部分相匹配的字体。

如果您需要的只是 8 像素的 charter 字体，可使用：

```
*-charter-*-*-*-*-8-*
```

查看选定的字体组属性

通过从"式样管理器字体"对话框中选择"属性"按钮，可以查看下面的字体组的属性：

- 字体组
- 大小
- 别名
- 别名 XLFD
- 别名位置
- 字体
- 字体 XLFD

用户的字体组文件系统存储

当用户添加字体组时，它被存储在下面的位置：

HomeDirectory/`.dt/sdtfonts`/*host*/*locale*/*typeface-nnnnnn*

host 是本地工作站的主机名

locale 是用户的当前语言环境，像 "C"或 "ja"。

typeface-nnnnnn 是由选中的字体和唯一生成的数字派生的名称

此字体组目录包含下面的三个文件

- `fonts.alias`
- `fonts.dir`
- `sdtfonts.group`

文件 `fonts.alias` 和 `fonts.dir` 是标准的 X11 字体文件，适于添加到 X 服务器的字体路径。`sdtfonts.group` 文件包含用户指定的字体组名称。

系统管理员字体组创建

这样其他的用户可以访问工作站上的字体组，系统管理员可将字体组复制到 /etc/dt/sdtfonts/*locale* 目录或复制到 /usr/openwin/lib/X11/stdfonts/*locale* 目录。“会话管理器”首先在 *HomeDirectory*/.dt/stdfonts/*host*/*locale* 中查找，接着在 /etc/dt/sdtfonts/*locale* 中查找，最后是在 /usr/openwin/lib/X11/stdfonts/*locale* 中查找。

管理颜色

本节描述：

- “式样管理器”如何设置显示颜色。
- “式样管理器”用于控制桌面颜色用法所使用的资源。

颜色调色板

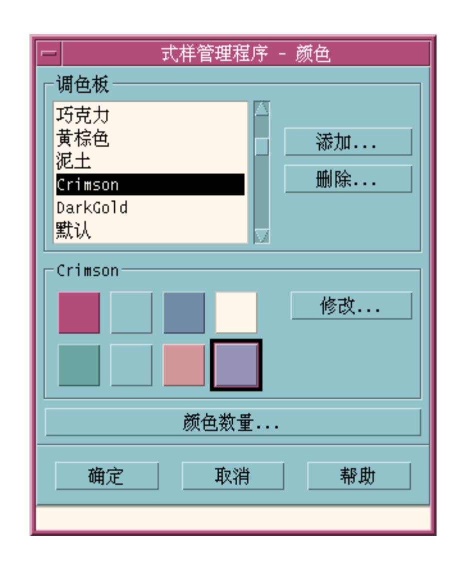
调色板由一组颜色集组成。当前调色板的颜色集显示在“式样管理器颜色”对话框中

每个调色板都有一个相对应的文件。paletteDirectories 资源 指定了包含调色板文件的目录。缺省情况下，此资源包含：

- 内置调色板：/usr/dt/palettes
- 系统范围调色板：/etc/dt/palettes
- 个人调色板：*HomeDirectory*/.dt/palettes

颜色集

在“式样管理器”的“颜色”对话框中，当前调色板中的每个颜色集由一个颜色按钮代表。每种颜色由一个颜色集 ID—从 1 到 8 的数字，来标识。



图形 17–1 HIGH_COLOR 的颜色集 ID 值

每个颜色集最多由五种颜色组成。每个颜色按钮显示颜色集的背景颜色。每个颜色集中的五种颜色代表下面的显示组件资源：

前景—应用程序窗口或窗框的前景。它总是黑色或白色。这一般是用于窗口和标题内的文本。

背景—应用程序或窗框的背景。

topShadowColor—应用程序控制 (如按钮) 和窗框架的顶部和左 bevel (斜面) 的颜色。

bottomShadowColor—应用程序控制和窗框架的底部和右 bevel (斜面) 的颜色。

selectColor—指示某些控制活动的活跃状态的颜色，象活动的切换和按钮。

每个调色板所使用的颜色集的数目取决于 `colorUse` 资源，用户可以使用"式样管理器要用颜色数目"对话框来设置其数目。

指定颜色值

在向其调色板文件写入颜色信息时，"式样管理器"使用 RGB 值。RGB 数的语法为：

RedGreenBlue

Red、*Green* 和 *Blue* 是十六进制数，每个 1 到 4 位长，这表示所使用的颜色的数目。每个颜色必须具有相同的位数。这样，有效的颜色值由 3、6、9 或 12 个十六 进制位组成。

例如，白色可用下面的任一方式来指定：

```
#fff
#ffffff
#fffffffff
#ffffffffffff
```

如果直接设置颜色资源，可使用颜色名称或 RGB 值。文件 /usr/lib/X11/rgb.txt 列出了全部已命名的颜色。

颜色集如何映射到资源

桌面通过资源将颜色集映射到各种显示单元，并作出如表 17–5所示的指定。

表 17–5 将颜色集映射到资源的

资源	显示单元
activeColorSetId	活动窗框颜色
inactiveColorSetId	非活动窗框颜色
textColorSetId	文本输入区
primaryColorSetId	应用程序的主背景区
secondaryColorSetId	应用程序的菜单条、菜单和对话框

这些资源将颜色集 ID 作为它们的值。当使用"式样管理器"选定新的调色板时，带有颜色集 IDs 的彩色显示单元允许单元动态地更改为新的颜色方案。

可将这些资源用于单个应用程序。例如，下行显示如何通过使用颜色集 8 作为窗口的主颜色，将所有的 dtterm 窗口可视地分组。

```
dtterm*primaryColorSetId:   8
```

缺省颜色集赋值

用于显示单元的颜色集 IDs 取决于在"式样管理器"中颜色设置的数目：

表 17–6 显示高保真彩色 (8 个颜色集) 的颜色集 IDs—"式样管理器"设置为"桌面使用更多颜色"。

表 17–6 高保真彩色

颜色集 ID	显示单元
1	活动窗框颜色
2	非活动窗框颜色

表 17–6 高保真彩色 (续下)

颜色集 ID	显示单元
3	未用 (缺省情况下)
4	文本输入区
5	应用程序的主背景区
6	应用程序的菜单条、菜单和对话框
7	缺省情况下未用
8	“前面板”背景

表 17–7 显示中某彩色 (4 个颜色集) 的颜色集 IDs—“式样管理器”设置为“应用程序使用更多颜色”。

表 17–7 中等彩色

颜色集 ID	显示单元
1	活动窗框颜色
2	非活动窗框颜色
3	“应用程序”和“前面板”背景颜色
4	文本输入区

表 17–8 显示低彩色 (2 个颜色集) 的颜色 IDs—“式样管理器”设置为“应用程序使用大多数颜色”。

表 17–8 低彩色

颜色集 ID	显示单元
1	活动窗框，工作区选择按钮
2	所有其他显示单元

用式样管理器控制颜色

可以通过“式样管理器”动态地更改桌面应用程序和其他合作应用程序的颜色。由“式样管理器”设置的前景和背景颜色可用于非合作应用程序。

对于这些客户程序而言，要响应“式样管理器”的颜色更改，客户机程序必须在使用桌面 Motif 库。用其他工具包写入的客户机程序无法动态地更改颜色来响应“式样管理器“的更改。 对于客户机程序的颜色更改只有在重新启动客户机程序后才生效。

必须不存在其它特别的颜色资源适用于客户程序。这包括指定用户的资源、appdefaults (应用缺省) 和在应用程序中内置的资源。

客户机程序可指定 primaryColorSetId 和 secondaryColorSetId 资源在桌面调色板中使用某些颜色。

式样管理器所用的颜色数目

“式样管理器”所使用的颜色数取决于下面的资源的值：

colorUse—配置桌面所使用的颜色数

shadowPixmaps—指示桌面用像素图来替换这两种阴影颜色

foregroundColor—指定前景颜色是否动态地更改

dynamicColor—当切换调色板时控制应用程序是否更改颜色

表 17–9 列出由桌面分配的最大颜色数。

表 17–9 桌面颜色的数目

显示	最大颜色数	数字采自
B_W	2	黑色和白色
LOW_COLOR	12	两个颜色集乘以五种颜色，再加上黑色和白色
MEDIUM_COLOR	22	四个颜色集乘以五种颜色，再加上黑色和白色
HIGH_COLOR	42	八个颜色集乘以五种颜色，再加上黑色和白色

要决定最大颜色数：

1. **将调色板中颜色集的数目与每个颜色集中的颜色数相乘。**
2. **加上 2 (用于黑色和白色)。**

   但是，在下面的配置情况下，在调色板中将只有十种颜色：四个颜色集乘以每个集中的两种颜色 (background 和 selectColor)，再加上黑色和白色：

```
*colorUse:    MEDIUM_COLOR
*shadowPixmaps:     True
*foregroundColor:    White
```

注意：多重颜色图标使用 14 种附加颜色。

---

colorUse 资源

colorUse 资源的缺省值是 MEDIUM_COLOR。此资源的值影响在调色板中使用的颜色集的数目。其他资源影响用于制造阴影的颜色数。colorUse 资源的值也影响多重颜色图标的使用。

值	描述
B_W	“黑色和白色”“式样管理器”将“显示”用 1 到 3 个颜色平面设置颜色集的数目：2 最大颜色数： 2 缺省颜色数：2 无多重颜色图标
LOW_COLOR	“应用程序使用大多数颜色”“式样管理器”将“显示”用 4 到 5 个颜色平面设置颜色集的数目：2 最大颜色数：12 缺省颜色数：12 无多重颜色图标
MEDIUM_COLOR	“应用程序使用更多颜色”“式样管理器”将“显示”用 6 个颜色平面设置颜色集的数目：4 最大颜色数：22 缺省颜色数：22 多重颜色图标
HIGH_COLOR	“桌面使用更多颜色”“式样管理器”将“显示”用 7 个或更多个颜色平面设置颜色集的数目：8 最大颜色数：42 缺省颜色数：42 多重颜色图标
缺省	桌面为显示选择正确的值。(为减少桌面用于高保真彩色的颜色数，缺省的 colorUse 资源设置为 MEDIUM_COLOR。)

shadowPixmaps 资源

shadowPixmaps 资源指示桌面用像素图来代替两个阴影颜色。这些像素图将黑色或白色与背景颜色相混合，以便模拟顶部或底部阴影值。这将需要的颜色数减少了两个，因为不需要为阴影颜色分配颜色单元。

值	描述
真	桌面创建 topShadowPixmap 和 bottomShadowPixmap 来代替阴影颜色使用。
假	从调色板使用 topShadowColor 和 bottomShadowColor。

shadowPixmaps 的缺省值取决于您的 colorUse 资源和显示器的硬件支持。

foregroundColor 资源

foregroundColor 资源指定在调色板中前景是如何配置的。

设置	结果
白色	前景设置为白色。
黑色	前景设置为黑色。
动态 (缺省)	前景被动态地设置为黑色或白色，取决于 background 的值。例如，白色字母在黄色背景上很难阅读，所以系统选择黑色。

如果 foregroundColor 被设置为黑色或白色，则颜色集中的颜色数将减少一个，并且前景将不随背景颜色的更改而更改。

foregroundColor 的缺省值是动态的，但在 colorUse 的值为 B_W 的地方例外。

dynamicColor 资源

dynamicColor 资源控制应用程序是否动态地更改颜色；就是说，当切换调色板时，客户机程序是否更改颜色。

值	描述
真	当选中新的调色板时，客户机程序动态地更改颜色。 这是缺省值。
假	客户机程序不动态地更改颜色。当选中了新的调色板时， 客户机程序将在会话重新启动时，使用新的颜色。

当 dynamicColor 资源的值为"真"时，不能动态地更改颜色的客户程序 (non-Motif 应用程序) 与能动态地更改颜色的客户程序在颜色颜色影射图中分配不同的单元，即使您看到相同的颜色。

注意：由于所有的客户机程序可以共享相同的颜色单元，将 dynamicColor 设置为"假"将减少桌面消耗的颜色数。

设置应用程序窗口的阴影厚度

桌面定义在应用程序窗口中的组件的缺省阴影厚度为一个像素，这些组件包括按钮阴影和焦点高亮显示。Motif 1.2 应用程序使用此资源值；其他应用程序可能未获得此资源值，因而在显示上将有所不同。

将 non-Motif 1.2 应用程序的阴影厚度设置为一个像素：

1. **以超级用户登录。**
2. **创建** `/etc/dt/config/`*language*`/sys.resources` **文件。**
3. **在** `/etc/dt/config/`*language*`/sys.resources` **中指定应用程序专用的资源如下：**

   application_class_name `*XmCascadeButton*shadowThickness: 1`

   有关为所有桌面用户覆盖系统缺省资源和指定附加资源的详细信息，请参见第 42 页"装入会话资源"

第 18 章

配置本地化的桌面会话

如果要配置本地化的桌面会话，需要：

- 设置 LANG 环境变量及其他“国家语言支持”(NLS) 环境变量
- 访问与语言相关的消息目录和资源文件
- 通过国际化系统，远程执行应用程序

管理 LANG 环境变量

要使用操作系统中对语言敏感的例程，就必须为桌面设置 LANG 环境变量。桌面支持：

- 西欧及拉丁语系语言
- 日语
- 繁体中文
- 简体中文
- 韩语

注意：桌面供应商可能已经添加了对其他类型语言的支持。

可以将 LANG 设置为操作系统所支持的任何值。 登录屏幕中的"选项"菜单显示了系统所支持的语言和地区列表。

为桌面设置 LANG 的方法有四种：

- 编辑 Xconfig 文件中的资源
- 使用登录屏幕中的"选项"菜单
- 创建可执行的 sh 或 ksh 的 Xsession.d 脚本。(有关使用 Xsession.d 脚本的详细信息，请参见第 39 页"查找 Xsession.d 脚本" 。)
- 通过编辑用户的 .dtprofile 文件

当设置完 LANG 后，桌面使用下面的与语言相关的文件来决定本地化接口。

颜色—/usr/dt/palettes/desc.*language*

背景—/usr/dt/backdrops/desc.*language*

为多个用户设置语言

如果通过 Xconfig 文件来设置语言，登录屏幕将本地化，并为所有用户设置了 LANG。该方法是在多重显示器系统中为所有显示器更改 LANG 的唯一方法。 (要修改 Xconfig，请将 /usr/dt/config/Xconfig 复制到 /etc/dt/config/Xconfig。)

设置语言的方法是，在 /etc/dt/config/Xconfig 文件中插入如下一行：

dtlogin.*host_display*.language: *language*

例如，下面一行将显示器 my_host:0 中的 LANG 设置为 Swedish_locale。

```
dtlogin.my_host_0.language: Swedish_locale
```

dtlogin 客户机程序读取该语言相应的消息目录，并调用本地化的登录屏幕。然后，dtlogin 客户机程序使用 /etc/dt/config/Xresources 资源文件中的如下资源来决定语言环境列表：

- dtlogin*language
- dtlogin*languageList
- dtlogin*languageName

Xconfig 文件可能需要为所选择的语言来适当设置 NLSPATH 环境变量。如果不是这种情况，或者希望自己来设置 NLSPATH，请参见第 277 页"NLSPATH 环境变量"。

设置一个会话的语言

要为一个会话设置语言，请使用登录屏幕"选项"菜单。登录屏幕将本地化，并为该用户设置了 LANG 。在会话结束时，LANG 将返回其缺省值 (在 dtlogin 中设置)。

设置一个用户的语言

用户可以在 *HomeDirectory*/.dtprofile 文件中覆盖登录的 LANG 设置。登录屏幕将不会本地化，而为用户设置 LANG 。

- 如果使用 sh 或 ksh：

```
LANG=language
 export LANG
```

- 如果使用 csh：

```
setenv LANG language
```

LANG 环境变量和会话配置

LANG 环境变量更改了在其中搜索会话配置文件的目录名。

本地化的会话的配置文件为：

- /usr/dt/config/*language*/Xresources (“登录管理器”资源文件)
- /usr/dt/config/*language* /sys.font (“会话管理器”资源文件)
- /usr/dt/config/*language* /sys.resources (“会话管理器”资源文件)
- /usr/dt/config/*language* /sys.session (“会话管理器”可执行 shell)
- /usr/dt/config/*language* /sys.dtwmrc (“窗口管理器”资源文件)
- /usr/dt/appconfig/types/*language* /dtwm.fp (“窗口管理器”前面板)

设置其他的 NLS 环境变量

除了 LANG 外，还有一些其他的 NLS 环境变量，例如 LC_CTYPE 和 LC_ALL. 。这些变量不受 dtlogin 语言资源或登录屏幕“选项”菜单的影响。 它们必须在下述文件中设置
：

- 系统范围变量： /etc/dt/config/Xsession.d
- 个人变量：*HomeDirectory* /.dtprofile

NLSPATH 环境变量

NLSPATH 环境变量决定了应用程序搜索消息目录的目录路径。必须将 LANG 和 NLSPATH 都设置为使用该消息目录。 有关本地化消息的位置的详细信息，请参考第 281 页“本地化信息目录”。大多数桌面客户机程序将预先指定 NLSPATH 启动时的路径。

查找字体

桌面包含的字体在 /usr/lib/X11/fonts 目录中。每个目录含一个目录文件 fonts.dir 和一个别名文件 fonts.alias。有关创建 fonts.dir 和 fonts.alias 文件的详细信息，请参见 mkfontdir 手册页。

要列出服务器上所有可用的字体，请使用 xlsfonts 命令。要添加或删除服务器的字体，请使用 xset 命令。

本地化 app-defaults 资源文件

桌面客户的 app-defaults 文件的缺省位置为 /usr/dt/app-defaults/*language*。例如，如果将 LANG 设置为 Swedish_locale，那么应用程序就会在 /usr/dt/app-defaults/Swedish_locale 中查找 app-defaults 文件。如果没有设置 LANG ，将忽略**语言** ，应用程序在 /usr/app-defaults/C 目录中寻找其 app-defaults 文件。

要更改 app-defaults 文件的位置，请使用 XFILESEARCHPATH 环境变量。例如，如果要将 app-defaults 移动到 /users 目录中，可将 XFILESEARCHPATH 变量设置为 /usr/app-defaults/*language*/*classname*。

如果将 XFILESEARCHPATH 设置在 *HomeDirectory*/.dtprofile 中，其值对运行的所有桌面和 X 客户机都适用。非客户机找不到它们的资源文件，除非将这些文件链接或复制到 XFILESEARCHPATH 指定的目录。

本地化操作和数据类型

注意： 要定制 /usr/dt/appconfig 目录中的文件，可在定制之前将该文件复制到 /etc/dt/appconfig目录中。

操作和数据类型定义文件的查找路径包括语言相关的目录：

- 个人：*HomeDirectory* /dt/types
- 系统范围：/etc/dt/appconfig/types/*language*
- 内置：/usr/dt/appconfig/types/*language*

“应用程序管理器”的配置文件的查找路径是：

- 个人：*HomeDirectory* /dt/appmanager
- 系统范围：/etc/dt/appconfig/appmanager/*language*
- 内置：/usr/dt/appconfig/appmanager/*language*
  该目录下的文件和目录名均被本地化。

本地化图标和位图

要本地化图标，可用“图标编辑器”编辑图标，并将它保存在：

```
/etc/dt/appconfig/icons/language
```

如果将它保存在其他目录中，请将 XMICONSEARCHPATH 环境变量设置为包含保存该图标的目录。 XMICONBMSEARCHPATH 环境变量控制着用于搜寻图标的路径。

本地化背景名+

背景的本地化是通过使用说明文件 (desc.*language* 和 desc.backdrops) 来实现的。无任何指定的本地化目录 (例如 /usr/dt/backdrops/*language*) 供背景文件使用。所有的语言环境使用相同的背景文件集，但是它们有各自的 desc.*language* 文件，其中含有背景的译名。

说明文件包含已翻译的背景名称的资源说明。例如：

```
 Backdrops*Corduroy.desc:
Velours Backdrops*DarkPaper.desc:
PapierKraft Backdrops*Foreground.desc:
AvantPlan
```

desc.*language* 文件用以检索关于 *language* 语言环境的背景说明，以便在“式样管理器”中显示背景。如果有描述说明，则它将在“式样管理器”的背景列表中显示。 否则，将使用背景文件名。

用户可以在 *HomeDirectory*/.dt/backdrops/desc.backdrops 文件中添加自己的背景说明。该文件用于在忽略语言环境的情况下，检索所有用户添加的背景的背景说明。

说明文件的查找路径是：

- 个人：*HomeDirectory* /.dt/backdrops/desc.backdrops
- 系统范围：/etc/dt/backdrops/desc.*language*

■ 内置：/usr/dt/backdrops/desc.*language*

本地化调色板名

调色板的本地化是通过使用说明文件 (desc.*language* 和 desc.palettes) 实现的。不存在特定的本地化目录 (例如 /usr/dt/palettes/*language*)。所有语言环境使用相同的调色板文件集，但是它们都有各自含有调色板的译名的 desc.palettes 文件。

说明文件含有有关调色板译名的资源说明。例如：

```
Palettes*Cardamon.desc:        Cardamone
Palettes*Cinnamon.desc:        Cannelle
Palettes*Clove.desc:           Brun
```

desc.*language* 文件用于检索调色板有关语言环境 *language* 的说明，以使在“式样管理器”的列表中显示调色板。如果有描述说明，它将显示在“式样管理器”的调色板列表中。 否则，将会使用调色板文件名。

用户可在 *HomeDirectory*/.dt/palettes/desc.palettes 文件中添加自己的调色板说明。该文件用于在忽略语言环境的情况下，检索所有用户添加的调色板的调色板说明。

该说明文件的查找路径为：

■ 个人：*HomeDirectory* /.dt/palettes/desc.palettes
■ 系统范围：/etc/dt/palettes/desc.*language*
■ 内置：/usr/dt/palettes/desc.*language*

本地化帮助卷

如果已本地化了帮助卷，必须将它存储在下面的目录之一中。系统使用所找到的第一个帮助卷。 这些目录是按下面顺序搜索的：

■ 个人：*HomeDirectory* /.dt/help
■ 系统范围：/etc/dt/appconfig/help/*language*
■ 内置：/usr/dt/appconfig/help/*language*

本地化信息目录

如果本地化了消息目录，将它存储到下面的目录中：

/usr/dt/lib/nls/msg/*language*。

这些目录中含有 *.cat 文件。

远程执行本地化了的桌面应用程序

可以在任何安装了类似的本地化桌面的远程执行主机上调用本地化桌面应用程序。当应用程序启动时，在调用它的主机上的与 NLS 相关的环境变量的值将传送到远程主机上。然而，这些环境变量并不包含任何主机信息。

重新设置键盘映射

如果出现意外的字符或行为，或者字符不能显示或键入，可能需要重新设置或安装键盘映射或者更改输入法。

输入法是由 LC_CTYPE、LANG 或 LC_ALL 环境变量决定的，或者由 -lang 选项指定的语言决定。

例如，如果用户希望在 POSIX shell 中以 C 语言环境打开终端：

```
LANG=C    dtterm
```

这个新的终端使用 C 语言环境，包括 C 的输入法和字体。如果正使用指定语言的键盘，那么输入法可能 不接受输入的任何扩展字符。当用指定语言的键盘使用 C 语言环境时，在调用终端之前用户需要将 LC_CTYPE (或 LANG 或 LC_ALL) 环境变量设置为相应的值。

例如，用德国键盘使用 C 语言环境，应键入：

```
LANG=C  LC_CTYPE=DeDE dtterm
```

如果 X 服务器已经重新设置，并且键映射已经初始化，您可以在服务器使用 xmodmap 命令重新设置适当的键盘映射。

附录 A

dtconfig(1) Man Page

```
NAME
     dtconfig - desktop configuration utility
SYNOPSIS
     dtconfig [-d |-e |-kill |-reset |-p |-inetd |-inetd.ow ]
DESCRIPTION
     Desktop configuration utility.  Integrates CDE with the
     operating system of the underlying platform.  System root
     login privilege is required to use dtconfig.
OPTIONS
     -d   Disables desktop auto-start feature.  At end of boot
          cycle, platform's native text based login mechanism
          will be used.
     -e   Enable's desktop auto-start feature.  Desktop login
     -kill
          Kill desktop (window based) login process and any user
          sessions associated with it.  Return control to
          system's native text based console.
     -reset
          Tell desktop (window based) login process to reread
 its
          configuration file to incorporate any changes.

     -p   Printer actions for any printer known to platform will

          be created if such print actions do not already exist
          in the platform's actions database.  This option is
          executed automatically at boot time if desktop auto-
          start has been enabled.
     -inetd
          Adds /usr/dt/bin daemons to the /etc/inetd.conf file.
          Specific CDE background daemon setup includes
          rpc.ttdbserverd (ToolTalk), rpc.cmsd (Calendar
          Manager), and dtspcd (subprocess control).  This
```

```
 -inetd
          option is called automatically by Solaris CDE package
          installs.  This -inetd option is also useful for CDE
          daemon setup outside of normal Solaris CDE install,
          including system setup where /usr/dt has simply been
          mounted from some remote fileserver exporting the
          /usr/dt directory.
     -inetd.ow
          Switches the ToolTalk and Calendar Manager daemons
          (rpc.ttdbserverd & rpc.cmsd) start lines in
          /etc/inetd.conf back to the older /usr/openwin/bin
          area.  This option is called automatically by Solaris
          CDE package remove scripts when needed.  It is also
          useful outside of normal Solaris package remove opera-

          tions when /usr/dt is about to be manually removed or
          unmounted.
RETURN VALUES
     0    Successful completion
    >0    Error condition
FILES
     /usr/dt/bin/dtconfig       location of dtconfig utility
SEE ALSO
     dtlogin (1), dtprintinfo (1)
```

索引

数字和符号

A

B

C

D

E

U

V

W

X

安

按

帮

背

查

窗

创

错

打

当

登

电

调

定

搜

提

图

位

文

阴

隐

应

用

优

邮

语

远

在

执

只

终

主

组